MICROSOFT®
GW-BASIC® Interpreter

Manuale dell'utente

MICROSOFT®
GW-BASIC® Interpreter

Manuale dell'utente

Interprete Microsoft® GW-BASIC®

Manuale dell'utente

Microsoft Corporation

Indice

Figure

Tabelle

1 Benvenuti in GW-BASIC

Microsoft® GW-BASIC® è un semplice linguaggio di programmazione per computer, facile da imparare e da usare, che fa uso di istruzioni di derivazione inglese e notazioni matematiche. Con GW-BASIC si possono scrivere programmi semplici e complessi da eseguire sul computer. E' anche possibile modificare il software esistente scritto in BASIC.

Il manuale ha lo scopo di aiutare ad usare l'interprete GW-BASIC con il sistema operativo MS-DOS.

1.1 Configurazione di sistema richiesta

Questa versione di GW-BASIC richiede una versione di MS-DOS 3.2 o successiva.

1.2 Premesse

I file di GW-BASIC si trovano nel dischetto di MS-DOS posto nel retro della *Guida di riferimento di MS-DOS*. Ricordarsi di fare una copia di lavoro del dischetto prima di continuare.

Avvertenza Il manuale è scritto per gli utenti che hanno familiarità con il sistema operativo MS-DOS. Per maggiori informazioni sull'uso di MS-DOS, fare riferimento alla *Guida introduttiva* e alla *Guida di riferimento di Microsoft MS-DOS 3.2*.

1.3 Notazioni convenzionali

Nel manuale, per distinguere gli elementi di testo, sono usate le seguenti convenzioni:

grassetto	Usato per evidenziare termini e concetti molto importanti e per alcune opzioni e parametri.
corsivo	Usato per porzioni di testo da digitare o che appaiono sullo schermo e per parametri variabili contenuti nelle istruzioni.
`monospazio`	Usato per gli esempi di righe di comando, i codici e gli esempi di programma ed altri esempi.
PICCOLO MAIUSCOLO	Usato per tasti e sequenze di tasti.

Le parentesi quadre racchiudono gli elementi opzionali della riga di comando.

1.4 Organizzazione del manuale

Il *Manuale dell'utente di GW-BASIC* è diviso in sei capitoli, nove appendici ed un glossario:

Il capitolo 1, "Benvenuti in GW-BASIC", descrive il manuale.

Il capitolo 2, "Apprendimento di GW-BASIC", è una guida elementare per iniziare a programmare.

Il capitolo 3, "Revisione e pratica di GW-BASIC", permette di praticare i principi di GW-BASIC spiegati nel capitolo 2.

Il capitolo 4, "L'editor di GW-BASIC", tratta i comandi di editing da usare quando si crea o modifica un programma di GW-BASIC. Vengono anche spiegate anche le proprietà uniche dei dieci tasti di funzione ridefinibili e degli altri tasti e combinazioni di tasti.

Il capitolo 5, "La creazione e l'uso dei file", insegna come creare i file e usare le procedure input/output (I/O).

Il capitolo 6, "Costanti, variabili, espressioni e operatori", definisce gli elementi di GW-BASIC ed indica come usarli.

L'appendice A, "I codici ed i messaggi d'errore", è un sommario di tutti i codici e messaggi d'errore che si potrebbero incontrare in GW-BASIC.

L'appendice B, "Le funzioni matematiche", descrive come calcolare alcune funzioni matematiche che non sono intrinseche a GW-BASIC.

L'appendice C, "I codici di carattere ASCII", elenca i codici di carattere ASCII riconosciuti da GW-BASIC.

L'appendice D, "Le subroutine in linguaggio di assemblaggio", indica come includere subroutine in linguaggio di assemblaggio con GW-BASIC.

L'appendice E, "Conversione di programmi BASIC in GW-BASIC", fornisce indicazioni per convertire in GW-BASIC i programmi scritti in BASIC.

L'appendice F, "Comunicazioni", descrive le istruzioni di GW-BASIC richieste per supportare le comunicazioni asincrone RS-232 con altri computer e periferiche.

L'appendice G, "Equivalenti esadecimali", elenca gli equivalenti decimali e binari dei valori esadecimali.

L'appendice H, "I codici di tasto", elenca ed illustra i valori dei codici di tasto usati in GW-BASIC.

L'appendice I, "Caratteri riconosciuti da GW-BASIC", descrive il set di caratteri di GW-BASIC.

Il "Glossario" definisce parole e espressioni usate frequentemente in GW-BASIC.

2 Apprendimento di GW-BASIC

Questo capitolo descrive come caricare GW-BASIC nel sistema. Spiega anche i due differenti tipi di modalità operative, i formati di riga e gli elementi vari di GW-BASIC.

2.1 Caricamento di GW-BASIC

Per usare il linguaggio GW-BASIC, lo si deve caricare nella memoria del computer dalla copia di lavoro del dischetto di MS-DOS. Seguire la procedura seguente:

1. Accendere il computer.
2. Inserire la copia di lavoro del dischetto di MS-DOS nell'unità disco A del computer e premere RITORNO.
3. Digitare il seguente commando dopo il sollecito *A>* e premere RITORNO:
   ```
   gwbasic
   ```

Una volta caricato GW-BASIC, il prompt di GW-BASIC, *Ok*, rimpiazzerà il prompt di MS-DOS, *A>*.

Sullo schermo, la riga *XXXXX BYTE LIBERI* indica quanti byte sono disponibili nella memoria per GW-BASIC.

La descrizione dei compiti dei tasti di funzione (F1-F10) appare sulla riga di fondo dello schermo. Questi tasti di funzione possono essere usati per risparmiare tempo di digitazione. Il capitolo 4 "L'editor di GW-BASIC" comprende informazioni dettagliate sui tasti di funzione.

2.2 Modalità operative

Una volta inizializzato (caricato), GW-BASIC visualizza il prompt *Ok. Ok* significa che GW-BASIC è al **livello di comando**, cioè è pronto ad accettare comandi. A questo punto, GW-BASIC può essere usato in due modalità: **modalità diretta** o **modalità indiretta**.

2.2.1 Modalità diretta

In modalità diretta, le istruzioni ed i comandi di GW-BASIC sono eseguiti immediatamente dopo la digitazione. I risultati di operazioni aritmetiche e logiche sono visualizzati immediatamente e/o memorizzati per successivo uso, mentre le istruzioni vengono eliminate dopo l'esecuzione. Questa modalità è utilizzata per la messa a punto dei programmi e quando si desidera usare GW-BASIC come una calcolatrice per calcoli rapidi che non richiedono un intero programma.

2.2.2 Modalità indiretta

La modalità indiretta è usata per creare i programmi. Le righe di programma sono precedute sempre dai numeri di riga e sono memorizzate. Il programma memorizzato può essere eseguito digitando il comando RUN.

2.3 Il formato della riga di comando di GW-BASIC

La riga di comando di GW-BASIC permette di cambiare l'ambiente o le condizioni mentre si usa GW-BASIC.

Avvertenza Quando si specificano modifiche dell'ambiente operativo di GW-BASIC, mantenere l'ordine dei parametri mostrato nella sintassi dell'istruzione. Per saltare un parametro, inserire una virgola, in modo da indicare al computer che non si intende modificare quel parametro particolare.

GW-BASIC usa una riga di comando della seguente forma:

gwbasic[*nomefile*][<*instd*][[>]>*outstd*][/**f**:*n*][/**i**][/**s**:*n*][/**c**:*n*][/**m**:[*n*][,*n*]][/**d**]

il *nomefile* è il nome di un file programma di GW-BASIC. Se questo parametro è presente, GW-BASIC procede come se gli fosse stato assegnato un comando RUN. Se non è indicata l'estensione del nome del file, viene utilizzata l'estensione predefinita .BAS. L'estensione .BAS indica che il file è un file di GW-BASIC. Un nome di file può essere composto da un massimo di otto caratteri, a cui può essere aggiunta un'estensione (tre caratteri) separata dal nome con un punto.

<instd sostituisce all'input standard di GW-BASIC il file specificato. Quando usato, deve apparire prima di ogni parametro.

Potrebbe essere usato quando si hanno più file potenzialmente utilizzabili dal programma e si desidera specificare un file di input particolare.

>outstd reindirizza l'output standard di GW-BASIC (lo schermo) al file o alla periferica specificati. Quando usato, deve apparire prima di ogni parametro. L'uso di >> prima di *stdout* fa sì che l'output sia aggiunto, invece di sostituire il contenuto dell'unità di destinazione.

Per reindirizzare l'input standard (tastiera) o l'output standard (schermo), digitare una riga di comando nella forma seguente:

gwbasic *programma <fileinput*[>]>*fileoutput*

I parametri appaiono frequentemente nelle righe di comando. Essi designano un'azione specifica per il comando, diversa da quella abituale. Ogni parametro è preceduto da uno slash o barra (/).

/f:*n* imposta il numero massimo dei file che si possono aprire simultaneamente durante l'esecuzione di un programma di GW-BASIC. Ogni file richiede 194 byte per il Blocco di Controllo File (FCB), più 128 byte per il buffer di dati. La misura del buffer di dati può essere modificata con il parametro /s:. Se il parametro /f: è omesso, il numero massimo dei file aperti è impostato a 3. Questo parametro è ignorato se non viene specificato unitamente al parametro /i nella riga di comando.

/i fà in modo che GW-BASIC assegni staticamente lo spazio richiesto per le operazioni di file, sulla base dei parametri **/s** e **/f**.

/s:*n* imposta la lunghezza di record massima permessa per i file. L'opzione che specifica la lunghezza di record nell'istruzione OPEN non può oltrepassare questo valore. Se il parametro /s: è omesso, la lunghezza di record viene impostata automaticamente a 128 byte. La misura di record massima consentita è 32767.

/c:*n* controlla le comunicazioni RS-232. Se sono presenti schede RS-232, **/c:**0 disattiva il supporto RS-232 ed ogni successivo tentativo di I/O per ogni scheda RS-232 presente. Se il parametro **/c:** viene omesso, sono assegnati 256 byte per il buffer di ricezione e 128 byte per il buffer di trasmissione, per ogni scheda presente.

Il parametro **/c:** non ha alcun effetto in mancanza di schede RS-232. Il parametro **/c:***n* assegna *n* byte al buffer di ricezione e 128 byte al buffer di trasmissione per ogni scheda RS-232 presente.

/m:*n*[,*n*] imposta la posizione di memoria massima (prima *n*) e la dimensione di blocco massima (secondo *n*) usate da GW-BASIC. GW-BASIC tenta normalmente di assegnare 64K byte di memoria per dati e segmenti di stack. Se si intendono usare subroutine in linguaggio macchina con programmi di GW-BASIC, usare il parametro **/m:** per stabilire la posizione più alta che GW-BASIC può usare. La dimensione di blocco massima è un multiplo di 16 ed è usata per riservare ai programmi dell'utente (le subroutine in linguaggio di assemblaggio) uno spazio superiore a quello di lavoro di GW-BASIC.

Il valore prestabilito per la dimensione di blocco massima è la posizione di memoria più alta. Il valore prestabilito per la posizione di memoria più alta è 64K byte, a meno che venga specificata la dimensione di blocco massima, nel qual caso lo standard è appunto la dimensione di blocco massima.

/d permette a certe funzioni di restituire risultati a precisione doppia. Quando viene specificato il parametro **/d**, sono usati approssimativamente 3000 byte addizionali di spazio di codice. Le funzioni influenzate sono ATN, COS, EXP, LOG, SIN, SQR e TAN.

Avvertenza Tutti i numeri di parametro possono essere specificati in termini decimali, ottali (preceduti da **&O**) o esadecimali (preceduti da **&H**).

Le righe di comando di GW-BASIC sono analoghe alle seguenti:

Questa riga usa 64K byte di memoria e tre file; carica ed esegue il file programma SALARI.BAS:

```
A>gwbasic salari
```

Il seguente comando usa 64K byte di memoria e sei file; carica ed esegue il file programma INVENT.BAS:

```
A>gwbasic invent /F:6
```

Il seguente comando disattiva il supporto RS-232 ed usa solo i primi 32K byte di memoria. I 32K byte superiori sono riservati per i programmi dell'utente:

```
A>gwbasic /C:O /M:32768,4096
```

Il seguente comando usa quattro file e permette una lunghezza di record massima di 512 byte:

```
A>gwbasic /F:4 /S:512
```

Il seguente comando usa 64K byte di memoria e tre file. Assegna 512 byte ai buffer di ricezione RS-232 e 128 byte ai buffer di trasmissione e carica ed esegue il file programma TTY.BAS:

```
A>gwbasic TTY /C:512
```

Per maggiori informazioni circa le comunicazioni RS-232, vedere l'Appendice F.

Come reindirizzare input e output standard

Quando reindirizzate, le istruzioni INPUT, LINE INPUT, INPUT$ e INKEY$, sono lette dal file di input specificato invece che dalla tastiera.

Tutte le istruzioni PRINT scrivono al file di output specificato invece che allo schermo.

I messaggi d'errore vanno all'output standard e allo schermo.

L'input da KYBD: continua ad essere letto dalla tastiera.

L'output a SCRN: continua ad apparire sullo schermo.

GW-BASIC continua ad eseguire il trapping dei tasti se è stata usata l'istruzione *n* ON KEY.

Se si digita CTRL-BREAK con l'output reindirizzato, GW-BASIC chiude i file aperti e trasmette il messaggio `Interruzione alla riga xxxx` all'output standard, per poi concludersi e ritornare a MS-DOS.

Quando l'input è reindirizzato, GW-BASIC continua a leggere da esso fino a che incontra un CTRL-Z. Questa condizione può essere sottoposta a test con la funzione di fine file (EOF). Se il file non si conclude con un CTRL-Z, o se un'istruzione di input di file di GW-BASIC cerca di leggere oltre la fine di file, tutti i file aperti vengono chiusi e GW-BASIC ritorna a MS-DOS.

Per ulteriori informazioni su queste ed altre istruzioni, funzioni, comandi e variabili menzionate in questo testo, fare riferimento alla *Guida di riferimento*.

Ecco alcuni esempi di reindirizzo di input ed output.

```
GWBASIC PROGRAM >DATI.OUT
```

I dati letti da INPUT e LINE INPUT continuano a provenire dalla tastiera. I dati trasmessi da PRINT finiscono nel file DATI.OUT.

```
gwbasic PROGRAM <DATI.INP
```

I dati letti da INPUT e LINE INPUT provengono dal file DATI.INP. I dati trasmessi da PRINT continuano ad andare allo schermo.

```
gwbasic PROGRAM <INPUT.DAT >OUTPUT.DAT
```

I dati letti da INPUT e LINE INPUT provengono dal file INPUT.DAT e i dati trasmessi da PRINT finiscono nel file OUTPUT.DAT.

```
gwbasic PROGRAM <\VENDITE\CLAUDIO\TRASF.DAT
>>\VENDITE\VENDITE.DAT
```

I dati letti da INPUT e LINE INPUT provengono dal file \VENDITE\CLAUDIO\TRASF.DAT. I dati trasmessi da PRINT sono aggiunti al file \VENDITE\VENDITE.DAT.

2.4 Istruzioni, funzioni, comandi e variabili di GW-BASIC

Un programma di GW-BASIC è composto da diversi elementi: parole chiave, comandi, istruzioni, funzioni e variabili.

2.4.1 Parole chiave

Le parole chiave di GW-BASIC, come PRINT, GOTO e RETURN, hanno un significato speciale per l'interprete GW-BASIC. GW-BASIC interpreta le parole chiave come parte di istruzioni o comandi.

Le parole chiave sono chiamate anche **parole riservate**. Infatti, esse non possono essere usate come nomi di variabile, in quanto il sistema le interpreterebbe come comandi. Tuttavia, le parole chiave possono essere inserite nei nomi di variabile.

Le parole chiave sono memorizzate nel sistema come simboli (caratteri a 1 o 2 byte) per usare lo spazio di memoria il più efficientemente possibile.

2.4.2 Comandi

I comandi e le istruzioni sono entrambi istruzioni eseguibili. La differenza tra i comandi e le istruzioni è che i comandi sono eseguiti generalmente in modalità diretta o al livello di comando dell'interprete. Generalmente eseguono operazioni di editing e di caricamento o salvataggio di programmi.

2.4.3 Istruzioni

Un'istruzione, come ON ERROR...GOTO, è un gruppo di parole chiave di GW-BASIC, generalmente usato come parte di un programma. Quando il programma viene eseguito, le istruzioni sono poste in esecuzione nell'ordine e con le caratteristiche in cui appaiono.

2.4.4 Funzioni

L'interprete GW-BASIC esegue sia funzioni numeriche che alfanumeriche.

2.4.4.1 Funzioni numeriche

L'interprete GW-BASIC può eseguire alcuni calcoli matematici (aritmetici o algebrici). Ad esempio, calcola il seno, il coseno o la tangente di un angolo x.

Se non viene indicato diversamente, le funzioni numeriche restituiscono solo risultati in numeri interi o a precisione singola.

2.4.4.2 Funzioni alfanumeriche

Le funzioni alfanumeriche funzionano su stringhe. Ad esempio, TIME$ e DATE$ restituiscono l'ora e la data conosciute dal sistema.

2.4.4 3 Funzioni definite dall'utente

Le funzioni possono anche essere definite dall'utente per mezzo dell'istruzione DEF FN. Queste funzioni possono essere sia alfanumeriche che numeriche.

2.4.5 Variabili

Alcuni gruppi di caratteri alfanumerici, cui vengono assegnati di volta in volta valori diversi, sono chiamati **variabili**. Quando le variabili vengono inserite in un programma di GW-BASIC, forniscono informazioni nel corso dell'esecuzione.

Ad esempio, ERR definisce l'ultimo errore che si è verificato nel programma; ERL fornisce la posizione dell'errore. Le variabili possono essere anche definite e/o ridefinite dall'utente o dal contenuto del programma.

Tutti i comandi, istruzioni, funzioni e variabili di GW-BASIC sono descritti individualmente nella *Guida di riferimento*.

2.5 Il formato di riga

Ognuno degli elementi di GW-BASIC può andare a costituire sezioni di programma che sono chiamate **istruzioni**. Le istruzioni sono molto simili a frasi in lingua inglese e sono combinate in maniera logica per la creazione di programmi. La *Guida di riferimento* descrive tutte le istruzioni disponibili.

In un programma di GW-BASIC, le righe hanno il formato seguente:

nnnnn istruzione[istruzioni]

nnnnn è un numero di riga.

istruzione è un'istruzione di GW-BASIC.

Una riga di programma di GW-BASIC inizia sempre con un numero di riga e deve avere almeno un carattere (ma non più di 255). I numeri di riga indicano l'ordine in cui le righe di programma sono memorizzate e sono usati anche come riferimento quando si creano collegamenti o si procede a operazioni di modifica. La riga di programma termina quando si preme il tasto RITORNO.

Per effetto della logica del programma, la riga può contenere più di una istruzione. In questo caso, ogni istruzione deve essere separata da un segno di due punti (:). Ognuna delle righe nel programma dovrebbe essere preceduta da un numero di riga. Questo numero può essere qualsiasi numero intero da 0 a 65529. E' consuetudine usare come numeri di riga i multipli di 10 (10, 20, 30 etc.), in maniera da lasciare spazio per righe aggiuntive da inserire successivamente. Dal momento che il computer inserirà le istruzioni in ordine numerico, le righe aggiuntive non devono necessariamente apparire sullo schermo in ordine consecutivo: ad esempio, se anche si digitasse la riga 35 dopo la riga 60, il computer leggerebbe la riga 35 dopo la riga 30 e prima della riga 40. Questa tecnica evita di ridigitare un intero programma per includere una riga che era stata dimenticata.

La larghezza dello schermo è di 80 caratteri. Se l'istruzione oltrepassa questa larghezza, il cursore si sposterà automaticamente sulla successiva riga. Solo quando si preme il tasto RITORNO il computer riconosce la fine della riga. Occorre perciò evitare la pressione istintiva del tasto RITORNO quando si giunge al margine destro dello schermo. Il computer risolve il problema automaticamente. Un'alternativa valida è la pressione della combinazione CTRL-RITORNO, che fa muovere il cursore all'inizio della riga successiva senza effettivamente inserire un avanzamento di riga. Quando si preme RITORNO, l'intera riga logica (composta da più righe fisiche) è memorizzata da GW-BASIC come unica.

In GW-BASIC, qualsiasi riga di testo che inizia con un carattere numerico è considerata una riga di programma ed è eseguita in uno dei tre modi sotto illustrati, in seguito alla pressione del tasto RITORNO:

- Viene aggiunta una riga nuova al programma. Ciò avviene se il numero di riga è legale (nel campo da 0 a 65529) e se almeno un carattere alfabetico o speciale segue il numero di riga.
- Viene modificata una riga esistente. Ciò avviene se il numero specificato coincide con quello di una riga preesistente nel programma. Quest'ultima è sostituita con il testo della nuova riga inserita.

  Avvertenza Il riutilizzo di un numero di riga esistente fa perdere tutte le informazioni presenti nella riga originale. Porre particolare attenzione nella digitazione di numeri in modalità indiretta. Si rischia infatti la cancellazione accidentale di righe di programma.

- Viene eliminata una riga esistente. Ciò avviene se il numero specificato coincide con il numero di una riga preesistente e la nuova riga non specifica altro che il numero stesso. Se si cerca di eliminare una riga inesistente, viene visualizzato il messaggio d'errore `Numero di riga non definito.`

2.6 Come ritornare a MS-DOS

Prima di ritornare a MS-DOS, **si deve salvare** quanto inserito in GW-BASIC (che andrebbe in caso contrario perduto).

Per ritornare a MS-DOS, digitare quanto segue al prompt *Ok* e premere RITORNO:

```
system
```

Il sistema ritorna a MS-DOS ed il prompt *A>* appare sullo schermo.

3 Revisione e pratica di GW-BASIC

Le sessioni pratiche in questo capitolo aiuteranno a memorizzare e rivedere ciò che si è imparato.

3.1 Esempio per la modalità diretta

Si può usare il computer in modalità diretta per eseguire operazioni matematiche fondamentali. GW-BASIC riconosce i simboli seguenti come operatori aritmetici:

Operazione	*Operatore di GW-BASIC*
Addizione	+
Sottrazione	-
Moltiplicazione	*
Divisione	/

Per inserire un problema, si risponde al prompt *Ok* con un punto interrogativo (?), seguito dall'enunciazione del problema che si vuole risolvere e si preme il tasto RITORNO. In GW-BASIC, il punto interrogativo può essere usato intercambiabilmente con la parola chiave PRINT. La risposta viene visualizzata in entrambi i casi sullo schermo.

Digitare quanto segue e premere il tasto RITORNO:

```
?2+2
```

GW-BASIC visualizzerà la risposta sullo schermo:

```
?2+2
4
Ok
```

Per esercitarsi in altre operazioni matematiche, sostituire il segno (+) con l'operatore desiderato.

Il linguaggio GW-BASIC non si limita all'esecuzione di funzioni aritmetiche. Si possono eseguire anche funzioni complesse di algebra e trigonometria. I formati da utilizzare per queste funzioni sono descritti nel capitolo 6, "Costanti, variabili, espressioni e operatori".

3.2 Esempi per la modalità indiretta

Il linguaggio di GW-BASIC può essere usato per funzioni anche molto complesse. Si può creare un programma che esegue una serie di operazioni e ne visualizza il risultato finale. Per iniziare a programmare, si creano righe di programma chiamate **istruzioni.** Ricordare che ci può essere più di un'istruzione nella riga e che ogni riga è preceduta da un numero.

Ad esempio, per creare il comando PRINT 2+3, digitare quanto segue:

```
10 print 2+3
```

Quando si preme il tasto RITORNO, il cursore si sposta sulla riga successiva, senza che avvenga null'altro. Per fare eseguire il calcolo al computer, digitare quanto segue e premere il tasto RITORNO:

```
run
```

Lo schermo dovrebbe apparire così:

```
Ok
10 print 2+3
run
5
Ok
```

Quanto scritto è un programma in GW-BASIC.

Il computer non esegue il calcolo sino a quando lo si richiede esplicitamente (con il comando RUN). Ciò permette di inserire più righe di istruzioni.

Il seguente programma, ad esempio, è composto da due righe di istruzioni. Digitare:

```
10 x=3
20 print 2+x
```

Usare ora il comando RUN per indicare al computer di effettuare il calcolo.

Lo schermo dovrebbe apparire così:

```
Ok
10 x=3
20 print 2+x
run
5
Ok
```

Le due caratteristiche che distinguono un programma da un semplice calcolo sono:

1. le righe numerate
2. l'uso del comando RUN

Queste caratteristiche indicano al computer che tutte le istruzioni sono state digitate ed il calcolo può essere eseguito dal principio alla fine. La numerazione delle righe segnala al computer che si tratta di un programma e non di un semplice calcolo e che, conseguentemente, non deve eseguire le istruzioni sino a che viene digitato il comando RUN.

In altre parole, i calcoli sono eseguiti in modalità diretta, mentre i programmi sono scritti in modalità indiretta.

Per visualizzare di nuovo il programma, digitare il comando LIST e premere il tasto RITORNO:

```
list
```

Lo schermo dovrebbe apparire così:

```
Ok
10 x=3
20 print 2+x
run
Ok
5
Ok
list
10 X=3
20 PRINT 2+X
Ok
```

Si osserverà un piccolo cambio nel testo del programma: le lettere minuscole inserite sono state convertite in lettere maiuscole. Il comando LIST introduce infatti questo cambio automaticamente.

3.3 Tasti di funzione

I tasti di funzione sono tasti che sono stati assegnati a comandi usati frequentemente. I tasti delle dieci funzioni sono normalmente situati sul lato sinistro della tastiera. Un elenco di questi tasti e dei comandi loro assegnati appare in fondo allo schermo di GW-BASIC. Per risparmiare tempo di digitazione, si può premere un tasto di funzione invece di digitare il nome di un comando.

Ad esempio, per elencare di nuovo tutte le righe di programma, non è necessario digitare il comando LIST; si può invece usare il relativo tasto di funzione:

1. Premere il tasto F1.
2. Premere RITORNO.

Il programma dovrebbe apparire sullo schermo.

Per eseguire il programma, premere semplicemente il tasto F2, che è assegnato al comando RUN.

Con l'apprendimento di nuovi comandi, si imparerà anche il significato dei tasti da F3 a F10. Il capitolo 4, "L'editor di GW-BASIC", fornisce maggiori informazioni sui tasti usati in GW-BASIC.

3.4 Modifica delle righe

Ci sono due metodi fondamentali per modificare le righe. Si possono:

- eliminare e sostituire
- modificare direttamente con il comando EDIT

Per eliminare una riga, digitarne il numero e premere il tasto RITORNO. Ad esempio, se si digita *12* e si preme il tasto RITORNO, la riga numero 12 è eliminata dal programma.

Se si intende invece usare il comando EDIT, digitare *EDIT* seguito dal numero della riga che si vuole modificare. A titolo di esempio, digitare quanto segue e premere il tasto RITORNO:

```
edit 10
```

Si possono poi usare i tasti seguenti, per eseguire modifiche:

Tasto	*Funzione*
SU (↑) GIU' (↓) A SINISTRA (←) A DESTRA (→)	Muove il cursore all'interno dell'istruzione
BACKSPACE	Elimina il carattere alla sinistra del cursore
DEL	Elimina il carattere corrente
INS	Inserisce caratteri alla sinistra del cursore

Ad esempio, per modificare l'istruzione (riga) 10 in *x=4*, usare i tasti di direzione per muovere il cursore sotto il *3*, e digitare un 4. Il numero 4 sostituisce il numero 3 nell'istruzione.

Premere ora il tasto RITORNO e quindi il tasto F2.

Lo schermo visualizza quanto segue:

```
Ok
10 X=4
RUN
6
Ok
```

3.5 Come salvare i file programma

Creare un programma è come creare un file di dati. Il programma è un file che contiene istruzioni per il computer. Per potere usare di nuovo il programma, lo si deve salvare, come si farebbe con un file di dati.

Per salvare un file in GW-BASIC, si usa il seguente procedimento:

1. Premere il tasto F4.

   La parola di comando *SAVE"* appare sullo schermo.

2. Digitare un nome per il programma e premere il tasto RITORNO. Il file è salvato sotto il nome specificato.

Per richiamare un file salvato, usare il seguente procedimento:

1. Premere il tasto F3.

   Il comando *LOAD"* appare sullo schermo.

2. Digitare il nome del file.
3. Premere RITORNO.

Il file è caricato nella memoria e pronto per essere visualizzato, modificato o eseguito.

4 L'editor di GW-BASIC

Si possono modificare le righe di un programma GW-BASIC nella fase di inserimento o dopo la memorizzazione in un file programma.

4.1 Modifica di righe in file nuovi

Se si inserisce un carattere sbagliato all'atto della digitazione, lo si può eliminare con i tasti BACKSPACE o DEL, o con CTRL-H. Dopo che il carattere è stato eliminato, si può continuare a digitare nella riga.

Il tasto ESC elimina una riga in fase di digitazione. In altre parole, se non si è ancora premuto il tasto RITORNO e si desidera eliminare la corrente riga di programma, è sufficiente premere il tasto ESC.

Per eliminare l'intero programma che risiede correntemente in memoria, si inserisce il comando NEW. NEW è usato normalmente per liberare la memoria prima di inserire un nuovo programma.

4.2 Modifica di righe in file salvati

Dopo che si è inserito e salvato il programma, si può scoprire che sono necessari alcuni cambi. Per introdurre queste modifiche, usare innanzitutto l'istruzione LIST per visualizzare le righe desiderate:

1. Ricaricare il programma.
2. Digitare il comando LIST o premere il tasto F1.

3. Digitare il numero di riga o l'intervallo di righe da modificare.

Le righe prescelte appariranno nello schermo.

4.2.1 Modifica delle informazioni in una riga di programma

Si possono apportare modifiche alle informazioni contenute in una riga posizionando il cursore nel punto che si intende cambiare e procedendo in uno dei modi seguenti:

- Digitando sui caratteri che già ci sono.
- Eliminando i caratteri alla sinistra del cursore con il tasto BACKSPACE.
- Eliminando i caratteri all'altezza del cursore usando il tasto DEL.
- Inserendo i caratteri all'altezza del cursore mediante la pressione del tasto INS. I caratteri a destra del cursore si spostano a destra per fare spazio ai nuovi caratteri inseriti.
- Aggiungendo caratteri alla fine della riga di programma.

Se viene modificata più di una riga, premere RITORNO al termine di ogni riga. Le righe modificate saranno memorizzate nell'ordine numerico corretto, anche se le righe non sono aggiornate in tale ordine.

Avvertenza Le modifiche non vengono memorizzate sino a quando il tasto RITORNO viene premuto, con il cursore collocato in un qualsiasi punto della riga modificata.

Non è necessario spostare il cursore alla fine della riga prima di premere il tasto RITORNO. L'interprete GW-BASIC ricorda dove ogni riga finisce e trasferisce tutta la riga, anche se RITORNO è premuto mentre il cursore è situato nel centro o al principio della riga.

Per troncare una riga all'altezza del cursore, digitare CTRL-END o CTRL-E, e premere il tasto RITORNO.

Se si era salvato originalmente il programma in un file, assicurarsi di salvare la versione del programma modificata. Altrimenti, le modifiche non saranno registrate.

4.3 Tasti speciali

L'interprete GW-BASIC riconosce nove dei tasti numerici alla destra della tastiera. Riconosce anche il tasto BACKSPACE, il tasto ESC e il tasto CTRL. I tasti e le sequenze di tasti seguenti hanno funzioni speciali in GW-BASIC:

Tasto	*Funzione*
BACKSPACE o CTRL-H	Elimina l'ultimo carattere digitato o il carattere alla sinistra del cursore. Tutti i caratteri alla destra del cursore sono spostati di una posizione verso sinistra. I caratteri successivi e le righe all'interno della riga logica corrente sono spostati in alto come con il tasto DEL.
CTRL-BREAK o CTRL-C	Ritorna alla modalità diretta, senza salvare i cambi apportati alla riga corrente. Determina anche l'uscita dalla modalità di numerazione automatica impostata con AUTO.
CTRL-A SINISTRA o CTRL-B	Sposta il cursore all'inizio della parola precedente. La parola precedente è definita come il successivo carattere alla sinistra del cursore nel set da A a Z o nel set da 0 a 9.
CTRL-A DESTRA o CTRL-F	Sposta il cursore all'inizio della parola successiva. La parola successiva è definita come il successivo carattere alla destra del cursore nel set da A a Z o nel set da 0 a 9. In altre parole, il cursore si sposta al numero o lettera successivo dopo uno spazio vuoto o altro carattere speciale.
GIU' o CTRL-(-)	Sposta il cursore di una riga verso il basso.
A SINISTRA o CTRL-]	Sposta il cursore di una posizione a sinistra. Quando il cursore è giunto oltre il margine sinistro dello schermo, si sposta sul lato destro dello schermo, nella riga precedente.

A DESTRA O CTRL-\	Sposta il cursore di una posizione a destra. Quando il cursore è giunto oltre il margine destro dello schermo, si sposta sul lato sinistro dello schermo, nella riga seguente.
SU O CTRL-6	Sposta il cursore in alto di una riga.
CTRL-BACKSPACE O DEL	Elimina il carattere collocato sopra il cursore. Tutti i caratteri alla destra di quelli eliminati sono spostati a sinistra di una posizione. Se una riga logica si estende oltre una riga fisica, i caratteri sulle righe successive sono spostati a sinistra di una posizione per riempire lo spazio precedente ed il carattere nella prima colonna di ogni riga successiva è spostato in alto alla fine della riga precedente. DEL è l'opposto di INS. Eliminando testo si riduce la lunghezza della riga logica.
CTRL-END O CTRL-E	Cancella il testo dalla posizione del cursore alla fine della riga logica. Tutte le righe fisiche dello schermo sono cancellate sino a che si incontra RITORNO.
CTRL-N O END	Sposta il cursore alla fine della riga logica. I caratteri digitati poi da questa posizione sono aggiunti alla riga.
CTRL-RITORNO O CTRL-J	Sposta il cursore all'inizio della successiva riga (fisica) dello schermo. Ciò permette la creazione di righe logiche di programma di dimensione maggiore della larghezza fisica dello schermo. Le righe logiche possono essere lunghe fino a 255 caratteri. Questa funzione può anche essere usata per determinare un avanzamento di riga (linefeed).
CTRL-M O RITORNO	Inserisce una riga nel programma di GW-BASIC e sposta il cursore alla riga logica successiva.

CTRL-[O ESC	Cancella l'intera riga logica su cui è situato il cursore.
CTRL-G	Determina l'emissione di un beep dall'altoparlante del computer.
CTRL-K O HOME	Sposta il cursore sull'angolo superiore sinistro dello schermo. Il contenuto dello schermo non è modificato.
CTRL-HOME O CTRL-L	Libera lo schermo e colloca il cursore sull'angolo superiore sinistro.
CTRL-R O INS	Attiva e disattiva la modalità di inserimento. La modalità di inserimento è indicata dal cursore che evidenzia la metà inferiore della posizione del carattere. In modalità grafica (Graphics Mode), il cursore copre l'intera posizione del carattere. Quando la modalità d'inserimento è attivata, solo la metà inferiore della posizione del carattere è nascosta dal cursore. Quando la modalità d'inserimento non è attivata, i caratteri digitati sostituiscono i caratteri esistenti e la BARRA SPAZIATRICE cancella il carattere all'altezza del cursore e sposta il cursore stesso a destra di una posizione. Il tasto A DESTRA sposta il cursore a destra di una posizione, ma il carattere non viene eliminato. Quando la modalità di inserimento è disattivata, se si preme il tasto TABULAZIONE, il cursore si sposta sino a raggiungere la successiva posizione di tabulazione. Le posizioni di tabulazione sono stabilite ad ogni otto posizioni di carattere. Quando la modalità d'inserimento è attiva, i caratteri che seguono il cursore vengono spostati a destra mentre i caratteri digitati vengono inseriti all'altezza del cursore.

Ad ogni carattere digitato, il cursore si sposta di una posizione a destra. L'avanzamento di riga è mantenuto sotto controllo. Cioè, se i caratteri si spostano oltre il margine destro dello schermo, vengono inseriti a partire dalla sinistra su una riga successiva. Gli inserimenti incrementano la lunghezza della riga logica.

Quando la modalità d'inserimento è attiva, la pressione del tasto TABULAZIONE determina l'apertura di spazi vuoti dalla posizione corrente del cursore alla successiva posizione di tabulazione. Anche in questo caso, l'avanzamento di riga è controllato automaticamente.

CTRL-NUMLOCK O CTRL-S	Impone una pausa al computer. Per riprendere le operazioni, premere un qualsiasi tasto.
CTRL-PRTSC	Fa in modo che quanto stampato venga inviato alla stampante parallela (LPT1:). Premendo una seconda volta CTRL-PRTSC si arresta la trasmissione dei dati alla stampante.
SHIFT-PRTSC	Trasmette il contenuto corrente dello schermo alla stampante, creando in pratica una fotografia istantanea dello schermo.
CTRL-I O TAB	Sposta il cursore alla successiva posizione di tabulazione. Le posizioni di tabulazione sono inserite ad ogni otto colonne.

4.4 Tasti di funzione

Alcuni tasti o combinazioni di tasti fanno eseguire comandi o funzioni frequentemente usate riducendo i normali tempi di digitazione. Questi tasti sono chiamati **tasti di funzione**.

I tasti di funzione speciali posti normalmente sul lato sinistro della tastiera possono essere temporaneamente ridefiniti per rispondere ad esigenze di programmazione.

I tasti di funzione permettono l'inserimento di una voce di un massimo di 15 caratteri nel programma, premendo un solo tasto. I tasti di funzione sono classificati da F1 a F10. GW-BASIC ha funzioni speciali prestabilite per ognuno di questi tasti. Si noterà che la descrizione questi tasti di funzione appare sul fondo dello schermo di GW-BASIC. Tali assegnazioni sono state scelte sulla base di alcuni dei comandi usati più frequentemente.

Inizialmente, i tasti di funzione sono assegnati alle seguenti funzioni:

Tabella 4.1 Assegnazioni dei tasti di funzione di GW-BASIC

Tasto	*Funzione*	*Tasto*	*Funzione*
F1	LIST	F6	,"LPT1:"<-
F2	RUN<-	F7	TRON<-
F3	LOAD"	F8	TROFF<-
F4	SAVE"	F9	KEY
F5	CONT<-	F10	SCREEN 0,0,0,<-

Avvertenza Il segno <- dopo una funzione indica che non è necessario premere il tasto RITORNO dopo il tasto di funzione. Il comando selezionato viene immediatamente eseguito.

Se si preferisce, si possono cambiare le assegnazioni di questi tasti. Tutti i 10 tasti di funzione possono essere ridefiniti. Per maggiori informazioni, vedere le istruzioni KEY e ON KEY nella *Guida di riferimento*.

5 La creazione e l'uso dei file

Ci sono due tipi di file nei sistemi MS-DOS:

- **File programma,** che contengono il programma o le istruzioni per il computer
- **File di dati,** che contengono le informazioni usate per o create dai file programma

5.1 Comandi di file programma

Quelli contenuti nell'elenco seguente sono i comandi e le istruzioni più frequentemente usati con i file programma. Consultare la *Guida di riferimento* per informazioni più dettagliate.

SAVE *nomefile*[,**a**][,**p**]

Scrive sul disco il programma che risiede correntemente in memoria.

LOAD *nomefile*

Carica in memoria il programma da un disco. LOAD cancella il contenuto corrente della memoria e chiude tutti i file prima di caricare il programma.

RUN *nomefile*

Carica in memoria il programma da un disco e lo esegue immediatamente. RUN cancella il contenuto corrente della memoria e chiude tutti i file prima di caricare il programma.

MERGE *nomefile*

Carica in memoria il programma da un disco, senza cancellare il programma già in memoria.

KILL *nomefile*

Elimina il file specificato da un disco. Questo comando può anche essere usato con i file di dati.

NAME *vecchio nomefile* **AS** *nuovo nomefile*

Cambia il nome di un file su disco. Solo il *nomefile* viene cambiato. Il file non è modificato e rimane nello stesso spazio e posizione sul disco. Questo comando può essere usato anche con i file di dati.

5.2 File di dati

I programmi di GW-BASIC possono lavorare con due tipi di file di dati:

- File sequenziali
- File ad accesso casuale

I file sequenziali sono più facili da creare dei file ad accesso casuale, ma sono limitati in flessibilità e velocità quando si ha accesso ai dati. I dati scritti in un file sequenziale sono composti da una serie di caratteri ASCII. I dati sono memorizzati uno dopo l'altro (sequenzialmente) nell'ordine d'invio e sono letti nello stesso ordine.

La creazione e l'accesso ai file ad accesso casuale richiede un numero maggiore di operazioni, ma i file casuali occupano meno spazio sul disco, in quanto GW-BASIC li immagazzina nella forma di una stringa in un formato compresso.

Le sezioni seguenti discutono come creare ed usare questi due tipi di file di dati.

5.2.1 Creazione di un file sequenziale

Le istruzioni e funzioni seguenti sono usate con i file sequenziali:

CLOSE	LOF
EOF	OPEN
INPUT#	PRINT#
LINE INPUT#	PRINT# USING
LOC	UNLOCK
LOCK	WRITE#

Le operazioni di programma seguenti sono richieste per creare un file sequenziale ed avere accesso ai dati nel file:

1. Aprire il file in modalità output (O). Il programma corrente userà questo primo file per l'output:

   OPEN "O",#1,*"nomefile"*

2. Scrivere i dati sul file usando PRINT# o WRITE#:

   ```
   PRINT#1,A$
   PRINT#1,B$
   PRINT#1,C$
   ```

3. Per avere accesso ai dati nel file, si deve chiudere il file e riaprirlo in modalità input (I):

   CLOSE#1
   OPEN "I",#1,*"nomefile"*

4. Usare INPUT# o LINE INPUT# per leggere i dati dal file sequenziale nel programma:

   ```
   INPUT#1,X$,Y$,Z$
   ```

L'esempio 1 è un breve programma che crea un file sequenziale, DATI, utilizzando come input le informazioni provenienti dal terminale.

Esempio 1

```
10 OPEN "O",#1,"DATI"
20 INPUT "NOME";N$
30 IF N$="FINE" THEN END
40 INPUT "DIPARTIMENTO";D$
50 INPUT "ASSUNZIONE";A$
60 PRINT#1,N$;",","D$",";A$
70 PRINT:GOTO 20
RUN
NOME? NICOLETTA ROSSI
DIPARTIMENTO? MARKETING
ASSUNZIONE? 01/12/82
NOME? FABIO SANNINO
DIPARTIMENTO? AMMINISTRAZIONE
ASSUNZIONE? 12/03/85
NOME? DAVIDE VIGANO
DIPARTIMENTO? RICERCA
ASSUNZIONE? 04/03/78
NOME? LUCA FORESTI
DIPARTIMENTO? DIREZIONE
ASSUNZIONE? 21/12/78
NOME? FINE
OK
```

5.2.2 Accesso ad un file sequenziale

Il programma nell'esempio 2 ha accesso ai dati del file creato nel programma dell'esempio 1 e visualizza il nome di coloro che sono stati assunti nel 1978.

Esempio 2

```
10 OPEN "I",#1,"DATI"
20 INPUT#1,N$,D$,A$
30 IF RIGHT$(A$,2)="78" THEN PRINT N$
40 GOTO 20
50 CLOSE #1
RUN
DAVIDE VIGANO
LUCA FORESTI
INPUT dopo la fine alla riga 20
Ok
```

Il programma nell'esempio 2 legge, sequenzialmente, ogni articolo nel file. Quando tutti i dati sono stati letti, la riga 20 causa l'errore `Input dopo la fine`. Per evitare questo errore, inserire la riga 15, che usa la funzione EOF per verificare la fine del file:

```
15 IF EOF(1) THEN END
```

e sostituire la riga 40 con *GOTO 15*.

Un programma che crea un file sequenziale può scrivere anche dati formattati al disco con l'istruzione PRINT# USING. Ad esempio, l'istruzione seguente potrebbe essere usata per scrivere dati numerici nel disco senza espliciti caratteri di delimitazione:

```
PRINT#1,USING"####.##,";A,B,C,D
```

La virgola alla fine della stringa di formato serve per separare gli elementi nel file di disco.

La funzione LOC, quando usata con un file sequenziale, restituisce il numero di record di 128-byte che sono stati letti o scritti dal file da quando è stato aperto.

5.2.3 Aggiunta di dati ad un file sequenziale

Quando un file sequenziale è aperto in modalità O (output), il contenuto corrente viene distrutto. Per aggiungere dati ad un file esistente senza distruggerne il contenuto, occorre aprire il file in modalità A (aggiunta).

Il programma nell'esempio 3 può essere usato per creare o aggiungere dati ad un file chiamato NOMI. Questo programma illustra l'uso di LINE INPUT. LINE INPUT inserirà caratteri fino a che vede un indicatore di ritorno di riga (carriage return) o fino a che ha letto 255 caratteri. Questa istruzione non si ferma di fronte ad una virgola o a delle virgolette.

Esempio 3

```
10   ON ERROR GOTO 2000
20   OPEN "A",#1,"NOMI"
110  REM AGGIUNTA NUOVE VOCI AL FILE
120  INPUT "NOMI";N$
130  IF N$="" THEN 200 'Esce dal ciclo chiuso di input
140  LINE INPUT "INDIRIZZO? ";I$
150  LINE INPUT "COMPLEANNO? ";C$
160  PRINT#1,N$
170  PRINT#1,I$
180  PRINT#1,C$
190  PRINT:GOTO 120
200  CLOSE #1
2000 ON ERROR GOTO 0
```

Nelle righe 10 e 2000 è stata usata l'istruzione ON ERROR GOTO. Questa istruzione permette d'intrappolare l'errore e specifica la prima riga (2000) della subroutine di gestione degli errori. La riga 10 attiva la subroutine di gestione degli errori. La riga 2000 invece disattiva la subroutine di gestione degli errori ed è il punto a cui GW-BASIC si ricollega per stampare i messaggi d'errore.

5.3 File ad accesso casuale

Le informazioni nei file ad accesso casuale sono memorizzate e lette in unità distinte e numerate dette **record**. Dal momento che le informazioni sono richiamate da numeri, i dati possono essere richiamati da qualsiasi posizione del disco; non è necessario che il programma legga l'intero disco per trovare i dati, come quando si cercano informazioni in file sequenziali. GW-BASIC supporta file ad accesso casuale di grosse dimensioni. Il numero massimo di record logico è 2^{32}-1.

Le seguenti istruzioni sono usate con i file ad accesso casuale:

CLOSE	FIELD	MKI$
CVD	LOC	MKS$
CVI	LOCK	OPEN
CVS	LOF	PUT
EOF	LSET/RSET	UNLOCK
ET	MKD$	

5.3.1 Creazione di un file ad accesso casuale

Per creare un file di dati casuale, sono necessarie le seguenti operazioni:

1. Aprire il file per la modalità ad accesso casuale (R). Il seguente esempio specifica una lunghezza di record di 32 byte. Se la lunghezza di record è omessa, il valore prestabilito è di 128 byte.

   OPEN "R",#1,"*nomefile*",32

2. Usare l'istruzione FIELD per assegnare spazio nel buffer ad accesso casuale alle variabili che saranno scritte nel file:

   FIELD#1,20 AS N$,4 AS A$,8 AS T$

   In questo esempio, le prime 20 posizioni (byte) nel buffer ad accesso casuale sono assegnate alla variabile alfanumerica N$; le successive 4 posizioni sono assegnate a A$; le successive 8 a T$.

3. Usare LSET o RSET per spostare i dati nei campi del buffer ad accesso casuale nel formato con allineamento a sinistra o a destra (L=sinistra SET: R=destra SET). I valori numerici debbono essere trasformati in stringhe prima dell'inserimento nel buffer. MKI$ converte valori interi in stringhe, MKS$ converte valori a precisione singola e MKD$ converte valori a precisione doppia.

   LSET N$=X$

   LSET A$=MKS$(AMT)

   LSET T$=TEL$

4. Scrivere i dati dal buffer al disco usando l'istruzione PUT:

   PUT #1,CODICE%

Il programma nell'esempio 4 prende le informazioni di input (il terminale) e le scrive ad un file di dati ad accesso casuale. Ogni volta che l'istruzione PUT è eseguita, viene scritto un record al file. Nell'esempio, l'input di due caratteri CODE%, nella riga 30, diviene il numero di record.

Avvertenza Non usare una variabile alfanumerica dimensionata con FIELD in un'istruzione INPUT o LET. Ciò fa sì che il puntatore per quella variabile punti nello spazio di stringa invece che nel buffer ad accesso casuale.

Esempio 4

```
10 OPEN "R",#1,"INFOFILE",32
20 FIELD#1,20 AS N$, 4 AS A$, 8 AS T$
30 INPUT "CODICE A 2 CIFRE";CODE%
40 INPUT "NOMI";X$
50 INPUT "AMMONTARE";AMT
60 INPUT "TELEFONO";TEL$:PRINT
70 LSET N$=X$
80 LSET A$=MKS$(AMT)
90 LSET T$=TEL$
100 PUT #1,CODE%
110 GOTO 30
```

5.3.2 Accesso ad un file ad accesso casuale

Per avere accesso ad un file casuale, sono necessarie le seguenti operazioni:

1. Aprire il file in modalità R:

   OPEN "R",#1,"*nomefile*",32

2. Usare l'istruzione FIELD per assegnare spazio per le variabili che saranno lette dal file nel buffer casuale:

   FIELD, #1, 20 AS N$, 4 AS A$, 8 AS T$

   In questo esempio, le prime 20 posizioni (byte) nel buffer casuale di file sono assegnate alla variabile alfanumerica N$; le successive 4 posizioni sono assegnate a A$; le successive 8 a T$.

   Avvertenza In un programma che esegue sia INPUT che OUTPUT sullo stesso file ad accesso casuale, si può spesso usare una sola istruzione OPEN ed una sola istruzione FIELD.

3. Usare l'istruzione GET per spostare il record desiderato nel buffer ad accesso casuale:

   GET #1,CODE%

   I dati nel buffer possono ora essere letti dal programma.

4. Riconvertire i valori numerici in numeri usando le funzioni appropriate: CVI per numeri interi, CVS per valori a precisione singola e CVD per valori a precisione doppia.

   PRINT N$
   PRINT CVS(A$)
   .
   .
   .

Il programma nell'esempio 5 accede al file ad accesso casuale INFOFILE, che era stato creato nell'esempio 4. In seguito all'inserimento del codice a due cifre, le informazioni associate con quel codice vengono lette dal file e visualizzate.

Esempio 5

```
10 Open "R,#1,"INFOFILE",32
20 FIELD #1, 20 AS N$, 4 AS A$, 8 AS T$
30 INPUT "CODICE A DUE CIFRE";CODE%
40 GET #1, CODICE%
50 PRINT N$
60 PRINT USING "$$###.##";CVS(A$)
70 PRINT T$:PRINT
80 GOTO 30
```

Con i file ad accesso casuale, la funzione LOC restituisce il numero di record corrente. Il numero di record corrente è l'ultimo numero di record usato in un'istruzione GET o PUT. Ad esempio, la riga seguente termina l'esecuzione di programma se il numero di record corrente nel file #1 è più alto di 99:

```
IF LOC(1)>99 THEN END
```

L'esempio 6 è un programma d'inventario che illustra l'accesso a file ad accesso casuale. In questo programma, il numero di record è associato al codice del prodotto e si assume che l'inventario non contenga più di 100 differenti prodotti.

Le righe 900-960 inizializzano il file di dati scrivendo CHR$(255) come primo carattere di ogni record. Ciò è usato successivamente (riga 270 e riga 500) per determinare se una voce esiste già per quel prodotto.

Le righe 130-220 visualizzano le diverse operazioni (funzioni) di inventario eseguite dal programma. Quando si digita il numero di funzione desiderato, la riga 230 si ricollega alla subroutine appropriata.

Esempio 6

```
120 OPEN"R",#1,"INVEN.DAT",39
125 FIELD#1,1 AS F$,30 AS D$, 2 AS Q$,2 AS R$,4 AS T$
130 PRINT:PRINT "FUNZIONI:":PRINT
135 PRINT 1,"INIZIALIZZA FILE"
140 PRINT 2,"CREA NUOVA VOCE"
150 PRINT 3,"MOSTRA INVENTARIO DI UN PRODOTTO"
160 PRINT 4,"AGGIUNGI ALLO STOCK"
170 5,"SOTTRAI DALLO STOCK"
180 PRINT 6,"MOSTRA TUTTI I PRODOTTI SOTTO IL LIVELLO
DI RIORDINO"
220 PRINT:PRINT:INPUT"FUNZIONE";FUNZIONE
225 IF (FUNZIONE<1)or(FUNZIONE>6) THEN PRINT "NUMERO
FUNZIONE ERRATO":GOTO 130
230 ON FUCTION GOSUB 900,250,390,480,560,680
240 GOTO 220
250 REM CREA NUOVA VOCE
260 GOSUB 840
270 IF ASC(F$) < # 255 THEN INPUT"SOVRASCRIVI";A$: IF
A$ < # "S" THEN RETURN
280 LSET F$=CHR$(0)
290 INPUT "DESCRIZIONE";DESC$
300 LSET D$=DESC$
310 INPUT "QUANTITA' IN STOCK";Q%
320 LSET Q$=MKI$(Q%)
330 INPUT "LIVELLO DI RIORDINO";R%
340 LSET R$=MKI$(R%)
350 INPUT "PREZZO UNITARIO";P
360 LSET T$=MKS$(P)
370 PUT#1,PROD%
380 RETURN
390 REM MOSTRA VOCE
400 GOSUB 840
410 IF ASC(F$)=255 THEN PRINT "VOCE NULLA":RETURN
420 PRINT USING "CODICE PRODOTTO ###";PROD%
430 PRINT D$
440 PRINT USING "QUANTITA' DISPONIBILE #####";CVI(Q$)
450 PRINT USING "LIVELLO RIORDINO #####";CVI(R$)
460 PRINT USING "PREZZO UNITARIO L. ##.###";CVS(T$)
470 RETURN
480 REM AGGIUNGI ALLO STOCK
```

```
490 GOSUB 840
500 IF ASC(F$)=255 THEN PRINT "VOCE NULLA":RETURN
510 PRINT D$:INPUT "QUANTITA' DA AGGIUNGERE";A%
520 Q%=CVI (Q$)+A%
530 LSET Q$=MKI$(Q%)
540 PUT#1,PROD%
550 RETURN
560 REM SOTTRAI DALLO STOCK
570 GOSUB 840
580 IF ASC(F$)=255 THEN PRINT "VOCE NULLA":RETURN
590 PRINT D$
600 INPUT "QUANITA' DA SOTTRARRE";S%
610 Q%=CVI(Q$)
620 IF (Q%-S%)<0 THEN PRINT "SOLO";Q%;" NELLO STOCK"
:GOTO 600
630 Q%=Q%-S%
640 IF Q%= < CVI(R$) THEN PRINT "QUANTITA'
ATTUALE";Q%; "LIVELLO RIORDINO";CVI(R$)
650 LSET Q$=MKI$(Q%)
660 PUT#1,PROD%
670 RETURN
680 REM MOSTRA PRODOTTI SOTTO IL LIVELLO DI RIORDINO
690 FOR I=1 TO 100
710 GET#1,I
720 IF CVI(Q$)<CVI(R$) THEN PRINT D$;" QUANTITA'";
CVI(Q$) TAB(50) "LIVELLO RIORDINO";CVI(R$)
730 NEXT I
740 RETURN
840 INPUT "CODICE PRODOTTO";PROD%
850 IF(PROD% < 1)OR (PROD% # 100) THEN PRINT "CODICE
PRODOTTO ERRATO": GOTO 840 ELSE GET#1,PROD%:RETURN
890 END
900 REM INIZIALIZZA FILE
910 INPUT "CONFERMA";B$:IF B$ < # "S" THEN RETURN
920 LSET F$=CHR$(255)
930 FOR I=1 TO 100
940 PUT#1,I
950 NEXT I
960 RETURN
```

6 Costanti, variabili, espressioni ed operatori

Le informazioni in questo capitolo aiuteranno ad apprendere l'uso di costanti, variabili, espressioni e operatori in GW-BASIC per sviluppare programmi più sofisticati.

6.1 Costanti

Le costanti sono valori statici che l'interprete GW-BASIC usa nell'esecuzione del programma. Ci sono due tipi di costanti: costanti alfanumeriche e costanti numeriche.

Una **costante alfanumerica** è una sequenza di caratteri alfanumerici (da 0 a 255) racchiusi tra doppie virgolette. I seguenti sono esempi di costanti alfanumeriche:

"CIAO"
"Lit. 25000"
"Numero impiegati"

Le **costanti numeriche** possono essere positive o negative. Ci sono cinque tipi di costanti numeriche: **intere, a punto fisso, a punto mobile, esadecimali e ottali.**

Avvertenza GW-BASIC utilizza il formato di numeri anglosassone. Perciò il separatore decimale è il punto (.) ed il separatore delle migliaia è la virgola (,). In ogni caso, quando si digita un numero, non occorre digitare il separatore delle migliaia. Ad esempio, digitare *10000* e non *10,000*, né tantomeno *10.000*.

Cosianti	*Descrizione*
Intere	I numeri interi tra -32768 e +32767. Non contengono decimali.
A punto fisso	I numeri reali positivi o negativi che contengono punti decimali.
A punto mobile	I numeri positivi o negativi rappresentati in forma esponenziale. Una costante a punto mobile è composta da un numero intero o a punto fisso (la base), seguito dalla lettera E ed un altro numero intero (l'esponente). L'intervallo di variazione consentito per le costanti a punto mobile va da 3.0×10^{-38} a 1.7×10^{38}. Ad esempio: 235.988E-7 = .0000235988 2359E6 = 2359000000
Esadecimale	I numeri esadecimali con prefisso &H. Ad esempio: &H76 &H32F
Ottale	I numeri ottali con prefisso &O o &. Ad esempio: &O347 &1234

6.1.1 Forma a precisione singola e doppia

Le costanti numeriche possono essere numeri interi, numeri a precisione singola o a precisione doppia. Le costanti intere sono memorizzate solo come numeri interi. Le costanti numeriche a precisione singola sono memorizzate con 7 cifre (sebbene solo 6 possono essere precise). Le costanti numeriche a precisione doppia sono memorizzate con 17 cifre di precisione e stampate con un massimo di 16 cifre.

Una costante a precisione singola è una qualunque costante numerica con, alternativamente:

- sette o meno cifre
- una forma esponenziale che usa E
- un punto esclamativo (!) al termine del numero

Una costante a precisione doppia è una qualunque costante numerica con, alternativamente:

- otto o più cifre
- una forma esponenziale che usa D
- il simbolo (#) al termine del numero

I seguenti esempi presentano alcune costanti numeriche a precisione singola e doppia:

Costanti a precisione singola	*Costanti a precisione doppia*
46.8	345692811
-1.09E-06	-1.09432D-06
3489.0	4390.0#
22.5!	7654321.1234

6.2 Variabili

Le variabili sono i nomi che si sono scelti per rappresentare i valori in un programma di GW-BASIC. Il valore di una variabile può essere specificamente assegnato o può essere il risultato di calcoli nel programma. Se una variabile non ha un valore assegnato, GW-BASIC assume che il valore sia zero.

6.2.1 Nomi e dichiarazioni di variabile

I nomi di variabile di GW-BASIC possono essere di qualunque lunghezza; sono considerati un massimo di 40 caratteri. I caratteri permessi in un nome di variabile sono lettere, numeri e il punto decimale. Il primo carattere nel nome della variabile deve essere una lettera.

Sono permessi anche caratteri di dichiarazione di tipo.

Parole riservate (tutte le parole usate come comandi, istruzioni, funzioni e operatori di GW-BASIC) non possono essere usate come nomi di variabile. Tuttavia, tali parole possono essere inserite nel nome di variabile se accompagnate da altre lettere.

Le variabili possono rappresentare valori numerici o stringhe.

6.2.2 Caratteri di dichiarazione di tipo

I caratteri di dichiarazione di tipo indicano che cosa una variabile rappresenta. Sono riconosciuti i seguenti caratteri di dichiarazione di tipo:

Caratteri	*Tipo di variabile*
$	Variabile alfanumerica
%	Variabile intera
!	Variabile a precisione singola
#	Variabile a precisione doppia

I seguenti sono esempi di nomi di variabile per ogni tipo:

Tipo di variabile	*Esempio di nome*
Variabile alfanumerica	N$
Variabile intero	LIMITE%
Variabile a precisione singola	MINIMO!
Variabile a precisione doppia	PI#

Il tipo predefinito per un nome di variabile numerica è la precisione semplice. La precisione doppia, pur essendo molto precisa, usa più spazio di memoria e più tempo per il calcolo. La precisione semplice è sufficientemente precisa per la maggior parte di applicazioni. Tuttavia, la settima cifra significativa (se stampata) non sarà sempre precisa. Occorre fare molta attenzione quando si operano conversioni tra variabili intere, a precisione semplice e a precisione doppia.

La variabile seguente è un valore a precisione singola in quanto non è stato specificato nulla di diverso:

ABC

Le variabili che iniziano con FN sono considerate chiamate di funzioni definite dall'utente.

Le istruzioni di GW-BASIC DEFINT, DEFSTR, DEFSNG e DEFDBL possono essere incluse in un programma per dichiarare i tipi di valori per alcuni nomi di variabile.

6.2.3 Variabili di matrice

Una matrice è un gruppo o tabella di valori individuati dallo stesso nome di variabile. Ogni elemento in una matrice è caratterizzato da una **variabile di matrice**, che è un indice intero o una espressione intera. L'indice è racchiuso tra parentesi. Un nome di variabile di matrice ha un numero di indici correlato alla dimensione della matrice.

Ad esempio,

```
V (10)
```

individua un valore in una matrice a dimensione uno, mentre

```
T (1,4)
```

individua un valore in una matrice a dimensione due.

La dimensione massima per una matrice, in GW-BASIC, è 255. Il numero massimo di elementi per dimensione è 32767.

Avvertenza Se si usa una matrice con un indice più grande di 10, si deve usare l'istruzione DIM. Fare riferimento alla *Guida di riferimento* per maggiori informazioni. Se si usa un indice maggiore del massimo specificato, apparirà il messaggio d'errore `Dimensioni dell'indice errate`.

Le matrici multidimensionali (più indici separati da virgole) sono utili per memorizzare i dati di tabelle. Ad esempio, A(1,4) potrebbe essere usata per rappresentare una matrice di due righe e cinque colonne come la seguente:

Colonna	0	1	2	3	4
Riga 0	10	20	30	40	50
Riga 1	60	70	80	90	100

In questo esempio, A(1,2)=80 e A(0,3)=40.

Righe e colonne iniziano con 0, non 1, a meno che venga diversamente specificato. Per maggiori informazioni, vedere l'istruzione OPTION BASE nella *Guida di riferimento*.

6.2.4 Spazio di memoria necessario per memorizzare variabili

I diversi tipi di variabili richiedono porzioni di memoria differenti. Le considerazioni che seguono possono rivelarsi importanti in relazione a una limitata disponibilità di memoria o a programmi di grosse dimensioni.

Variabile	*Byte di memoria necessari*
Intera	2
A precisione semplice	4
A precisione doppia	8

Matrice	*Byte di memoria necessari*
Intera	2 per elemento
A precisione semplice	4 per elemento
A precisione doppia	8 per elemento

Stringhe

Tre byte più il contenuto della stringa (un byte per ogni carattere). Le virgolette che segnano l'inizio e la fine di ogni stringa non sono considerate.

6.3 Conversione di tipo

Quando necessario, GW-BASIC converte una costante numerica da un tipo di variabile ad un altro, secondo le regole seguenti:

- Se una costante numerica di un tipo è stabilita essere uguale ad una variabile numerica di differente tipo, il numero è memorizzato secondo il tipo dichiarato nel nome di variabile. Ad esempio:

```
10 A% = 23.42
20 PRINT A%
RUN
 23
```

Se una variabile alfanumerica è stabilita essere uguale ad un valore numerico o viceversa, si verifica l'errore `Tipi di carattere contrastanti.`

- Nel calcolo di una espressione, tutti gli operandi di una operazione aritmetica o di relazione sono convertiti allo stesso grado di precisione; cioè, quello dell'operando più preciso. Anche il risultato di un'operazione aritmetica è restituito a questo stesso grado di precisione. Ad esempio:

```
10 D# = 6#/7
20 PRINT D#
RUN
 .8571428571428571
```

Il calcolo aritmetico è eseguito in precisione doppia ed il risultato è restituito in D# come un valore a precisione doppia.

```
10 D = 6#/7
20 PRINT D
RUN
```

Il calcolo aritmetico è eseguito in precisione doppia ed il risultato è restituito a D (variabile a precisione singola) arrotondato e stampato come un valore a precisione singola.

- Gli operatori logici convertono i loro operandi in interi e restituiscono un risultato intero. Gli operandi debbono essere nell'intervallo di variazione -32768 / 32767.
- Quando un valore a punto mobile è convertito in un intero, la porzione frazionale è arrotondata. Ad esempio:

```
10 C% = 55.88
20 PRINT C%
RUN
 56
```

- Se ad una variabile a precisione doppia è assegnato un valore a precisione singola, sono valide solo le prime sette cifre (arrotondate) del numero convertito. Ciò accade perché solo sette cifre significative sono state fornite con il valore a precisione singola. Il valore assoluto della differenza tra il numero a precisione doppia stampato e il valore originale a precisione singola, è meno di 6.3^{-8} volte il valore originale a precisione singola.

Ad esempio:

```
10 A = 2.04
20 B# = A
30 PRINT A;B#
RUN
 2.04  2.039999961853027
```

6.4 Espressioni ed operatori

Un'espressione può essere semplicemente una costante numerica o alfanumerica, una variabile, oppure può combinare costanti e variabili con operatori per produrre un singolo valore.

Gli operatori eseguono operazioni matematiche e logiche sui valori. Gli operatori messi a disposizione da GW-BASIC sono divisi in quattro categorie:

- Aritmetici
- Relazionali
- Logici
- Di funzione

6.4.1 Operatori aritmetici

Operatori	*Operazioni*
^	Elevamento a potenza
-	Trasformazione segno
*	Moltiplicazione
/	Divisione a punto mobile
+	Addizione
-	Sottrazione

Le operazioni tra parentesi sono eseguite prima. All'interno delle parentesi è mantenuto il normale ordine di precedenza.

I seguenti sono esempi di espressioni algebriche. Per ognuna di esse viene indicato il formato da utilizzare in GW-BASIC:

Espressione algebrica	*Espressione BASIC*
$\frac{X-Y}{Z}$	(X-Y)/Z
$\frac{X*Y}{Z}$	X*Y/Z
$\frac{X+Y}{Z}$	(X+Y)/Z
$(X^2)^Y$	(X^2)^Y
X^{Y^Z}	X^(Y^Z)
X(-Y)	X*(-Y)

Due operatori consecutivi debbono essere separati da parentesi.

6.4.1.1 Divisione di interi e modulo aritmetico

Due altri operatori aritmetici possono essere utilizzati: la divisione di interi e il modulo aritmetico.

La divisione di interi è indicata dal backslash (\). Gli operandi sono arrotondati al valore intero (debbono rientrare nell'intervallo -32768 / 32767) prima che la divisione sia eseguita ed il quoziente è troncato in un intero (cioè arrotondato per difetto).

I seguenti sono esempi di divisione di interi:

```
10\4 = 2
25.68\6.99 = 3
```

In GW-BASIC, la divisione di interi sarà eseguita immediatamente dopo la divisione a punto mobile.

Il modulo aritmetico è indicato dall'operatore MOD. Dà il valore intero che rappresenta il resto di una divisione di interi.

I seguenti sono esempi di modulo aritmetico:

```
10.4 MOD 4 = 2
(10/4=2 con resto di 2)
25.68 MOD 6.99 = 5
(26/7=3 con resto di 5)
```

In GW-BASIC, il modulo aritmetico segue la divisione di interi. Le funzioni INT e FIX, descritte nella *Guida di riferimento*, sono usate anche nel modulo aritmetico.

6.4.1.2 Limiti matematici e divisione per zero

Se, durante il calcolo di una espressione, si incontra una divisione per zero, appare il messaggio d'errore `Divisione per zero`, viene restituito come risultato il simbolo d'infinito di macchina con il segno del numeratore e l'esecuzione continua.

Se si eleva zero ad una potenza negativa, appare il messaggio d'errore `Divisione per zero`, il simbolo d'infinito viene restituito come risultato del calcolo e l'esecuzione continua.

Se si superano i limiti matematici consentiti, appare il messaggio d'errore `Valore troppo elevato`, il simbolo d'infinito con il segno algebricamente corretto viene restituito come risultato del calcolo e l'esecuzione continua.

Nessuno degli errori elencati in questo paragrafo verrà "catturato" da una procedura di trapping degli errori.

6.4.2 Operatori relazionali

Gli operatori relazionali permettono di confrontare due valori. Il risultato del confronto è vero (-1) o falso (0). Questo risultato può essere poi usato per una decisione circa il flusso del programma.

La tabella 6.1 mostra gli operatori relazionali.

Tabella 6.1 Operatori relazionali

Operatori	*Relazione esaminata*	*Espressione*
=	Uguaglianza	X-Y
<>	Differenza	X<>Y
<	Minore	X<Y
>	Maggiore	X>Y
<=	Minore-uguale	X<=Y
>=	Maggiore-uguale	X>=Y

Il segno uguale è usato anche per assegnare un valore ad una variabile. Vedere l'istruzione LET nella *Guida di riferimento*.

Quando gli operatori aritmetici e di relazione sono combinati in una espressione, le operazioni aritmetiche sono sempre eseguite prima:

```
X+Y < (T-1)/Z
```

Questa espressione è vera se il valore di X sommato ad Y è minore del valore di (T-1) diviso per Z.

6.4.3 Operatori logici

Gli operatori logici eseguono test su relazioni multiple, manipolazioni di bit o operazioni booleane. L'operatore logico restituisce un risultato in bit che è vero (non zero) o falso (zero). In una espressione, le operazioni logiche sono eseguite dopo quelle aritmetiche e relazionali. Il risultato di una operazione logica è determinato come mostrato nella tabella seguente. Gli operatori sono elencati in ordine di precedenza.

Tabella 6.2 Risultati restituiti da operazioni logiche

Operazione	*Valore*	*Valore*	*Risultato*
NOT	X		NOT X
	V		F
	F		V
AND	X	Y	X AND Y
	V	V	V
	V	F	F
	F	V	F
	F	F	F

Operazione	*Valore*	*Valore*	*Risultato*
OR	X	Y	X OR Y
	V	V	V
	V	F	V
	F	V	V
	F	F	F
XOR	X	Y	X XOR Y
	V	V	F
	V	F	V
	F	V	V
	F	F	F
EQV	X	Y	X EQV Y
	V	V	V
	V	F	F
	F	V	F
	F	F	V
IMP	X	Y	X IMP Y
	V	V	V
	V	F	F
	F	V	V
	F	F	V

Proprio come gli operatori relazionali possono essere usati per decisioni circa il flusso del programma, gli operatori logici possono collegare due o più relazioni e restituire un valore vero o falso da usare in una decisione. Ad esempio:

```
IF D<200 AND F<4 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```

Gli operatori logici convertono i loro operandi in numeri di 16-bit, con segno, complemento di due, nell'intervallo -32768 / +32767. Se gli operandi non sono nell'intervallo specificato, si produce un errore.

Se entrambi gli operandi sono 0 o -1, gli operatori logici restituiscono 0 o -1. L'operazione data è eseguita su questi interi per bit; cioè, ogni bit del risultato è determinato dai bit corrispondenti nei due operandi.

E' così possibile usare gli operatori logici per esaminare byte con una particolare struttura di bit. Ad esempio, l'operatore AND può essere usato per mascherare tutti i bit di un byte di stato ad una porta di macchina I/O, tranne uno. L'operatore OR può essere usato per unire due byte al fine di creare un valore binario particolare.

I seguenti esempi dimostrano come funzionano gli operatori logici:

Esempio	*Spiegazione*
63 AND 16=16	63=binario 111111 e 16=binario 10000, perciò 63 AND 16=16
15 AND 14=14	15=binario 1111 e 14=binario 1110, perciò 15 AND 14=14 (binario 1110)
-1 AND 8=8	-1=binario 1111111111111111 e 8=binario 1000, perciò -1 AND 8=8
4 OR 2=6	4=binario 100 e 2=binario 10, perciò 4 OR 2=6 (binario 110)
10 OR 10=10	10=binario 1010, perciò 1010 OR 1010=1010 (10)
-1 OR -2=-1	-1=binario 1111111111111111 e -2=binario 1111111111111110, perciò -1 OR -2=-1. Il complemento di bit di 16 zeri è di 16 uno, che è poi la rappresentazione del complemento di due di -1.
NOT X=-(X+1)	Il complemento di due di qualunque intero è il complemento di bit più uno.

6.4.4 Operatori di funzione

Una funzione è usata in una espressione per richiamare un'operazione predeterminata che deve essere eseguita da un operando. GW-BASIC ha funzioni intrinseche, residenti nel sistema, come SQR (radice quadrata) o SIN (seno).

GW-BASIC permette anche all'utente di definire proprie funzioni. Si veda l'istruzione DEF FN nella *Guida di riferimento*.

6.4.5 Operatori di stringa

Per confrontare le stringhe, usare gli stessi operatori relazionali usati con i numeri:

Operatore	*Significato*
=	Uguale
<>	Diverso
<	Minore
>	Maggiore
<=	Minore-uguale
>=	Maggiore-uguale

L'interprete GW-BASIC confronta le stringhe prendendo un carattere alla volta da ogni stringa e confrontando i relativi codici ASCII. Se i codici ASCII in ogni stringa sono gli stessi, le stringhe sono uguali. Se i codici ASCII differiscono, il numero di codice più basso precederà il codice più alto. Se l'interprete raggiunge la fine di una stringa durante il confronto, la stringa più corta è considerata più piccola, se ambedue le stringhe sino a quel punto erano uguali. Spazi all'inizio o al termine di una parola sono significativi.

Ad esempio:

```
`"AA" < "AB"
"NOMEFILE" = "NOMEFILE"
"X&" > "X#"
"CL " > "CL"
"kg" > "KG"
"LUIS" < "LUISA
B$ < "9/12/78" where B$ = "8/12/78"
```

I confronti di stringhe possono essere anche usati per esaminare i valori alfanumerici o per alfabetizzare le stringhe. Tutte le costanti alfanumeriche usate in espressioni di confronto debbono essere racchiuse tra virgolette. Le stringhe possono essere concatenate usando il segno più (+), Ad esempio:

```
10 A$="FILE":B$-"NOMI"
20 PRINT A$+B$
30 PRINT "NUOVO " + A$+B$
RUN
NOMEFILE
NUOVO NOMEFILE
```

Appendice A
Codici e messaggi d'errore

Codice	*Messaggio*
1	`NEXT senza FOR`
	L'istruzione NEXT non ha una corrispondente istruzione FOR. Controllare la variabile di FOR e verificare se esiste una variabile di NEXT corrispondente.
2	`Errore di sintassi`
	Si è incontrata una riga che contiene una sequenza di caratteri errata (punteggiatura non corretta, errori di ortografia in un'istruzione, etc.). In seguito a questo errore, GW-BASIC visualizza la riga errata per la modifica.
3	`RETURN senza GOSUB`
	Si è incontrata un'istruzione RETURN per la quale non esiste una precedente istruzione GOSUB.
4	`DATA non presente`
	Si è eseguita un'istruzione READ, ma non esistono nel programma istruzioni DATA con dati non letti.
5	`Chiamata di funzione illegale`
	Un parametro fuori intervallo è stato passato ad una funzione matematica o alfanumerica. Una chiamata di funzione illegale può avvenire anche in seguito a:

- un indice negativo o eccessivamente grande
- un argomento negativo o uguale a zero con LOG
- un argomento negativo per SQR

- una base negativa con una potenza non intera
- la chiamata di una funzione USR il cui indirizzo di partenza non è ancora stato dato
- un argomento non adatto per MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR o ON...GOTO

6 `Valore troppo elevato`

Il risultato di un calcolo è troppo grande per essere rappresentato nel formato numerico di GW-BASIC. Se avviene il contrario (il risultato è troppo piccolo), il risultato fornito è zero e l'esecuzione continua senza errore.

7 `Memoria esaurita`

Il programma è troppo grande, ha troppi cicli di istruzioni FOR, GOSUB, variabili o espressioni molto complicate. Usare l'istruzione CLEAR per liberare spazio di stack o memoria.

8 `Numero di riga non definito`

Una riga di riferimento in un GOTO, GOSUB, IF-THEN...ELSE, o DELETE non esiste

9 `Dimensioni dell'indice errate`

Un elemento di matrice è richiamato con un indice che è oltre la dimensione della matrice o con un numero sbagliato di indici.

10 `Definizione doppia`

Sono state date due istruzioni DIM per la stessa matrice, o è stata data un'istruzione DIM per una matrice dopo che era stata impostata la dimensione standard 10 per quella matrice.

11 `Divisione per zero`

Si è incontrata una divisione per zero in una espressione oppure vi è uno zero elevato ad una potenza negativa. Il risultato dato è l'infinito di macchina con il segno del numeratore (positivo se si tratta di elevamento a potenza). L'esecuzione continua.

12 `Illegale in modalità diretta`

Un'istruzione non ammissibile in modalità diretta è stata inserita in tale modalità.

13	`Tipi di carattere contrastanti`
	E' stato assegnato un valore numerico ad un nome di variabile alfanumerica o viceversa. Oppure è stato dato un argomento alfanumerico ad una funzione che richiede un argomento numerico o viceversa.
14	`Memoria per stringhe esaurita`
	Variabili alfanumeriche hanno spinto GW-BASIC a superare la quantità di memoria libera rimasta. GW-BASIC assegna lo spazio di stringa dinamicamente sino a che si esaurisce la memoria.
15	`Stringa troppo lunga`
	Si è tentato di creare una stringa più lunga di 255 caratteri.
16	`Stringa troppo complessa`
	Un'espressione alfanumerica è troppo lunga o troppo complessa. Spezzare l'espressione in espressioni più piccole.
17	`Impossibile continuare`
	Si è tentato di continuare un programma che: • è bloccato a causa di un errore • è stato modificato durante una pausa nella fase di esecuzione • non esiste
18	`Funzione utente non definita`
	Una funzione USR è chiamata prima di essere definita come tale (istruzione DEF).
19	`Manca RESUME`
	E' stata inserita una routine di trapping di errori senza istruzione RESUME.
20	`RESUME senza errore`
	Vi è un'istruzione RESUME prima dell'inizio della routine di trapping di errori.
21	`Errore non riproducibile`
	Non è disponibile alcun messaggio d'errore per il tipo di errore verificatosi. Ciò è determinato generalmente da un errore con un codice indefinito.

22 `Operando mancante`

Un'espressione contiene un operatore privo di operandi.

23 `Buffer di riga pieno`

Si è tentato di inserire una riga composta da troppi caratteri.

24 `Timeout periferica`

GW-BASIC non ha ricevuto informazioni da una determinata unità periferica I/O per un determinato periodo di tempo.

25 `Errore nella periferica`

Indica un errore di hardware nella stampa o scheda d'interfaccia.

26 `FOR senza NEXT`

Vi è un FOR senza corrispondente NEXT.

27 `Senza carta`

La stampante non ha carta, oppure si è verificato un errore nella stampante.

28 `Errore non riproducibile`

Non è disponibile alcun messaggio d'errore per il tipo di errore verificatosi. Ciò è determinato generalmente da un errore con un codice indefinito.

29 `WHILE senza WEND`

Un'istruzione WHILE non ha un corrispondente WEND.

30 `WEND senza WHILE`

Un'istruzione WEND non ha un corrispondente WHILE.

31-49 `Errore non riproducibile`

Non è disponibile alcun messaggio d'errore per il tipo di errore verificatosi. Ciò è determinato generalmente da un errore con un codice indefinito.

50 `FIELD di dimensioni eccessive`

Si è tentato di assegnare, con un'istruzione FIELD, più byte di quelli specificati per la lunghezza del record di un file ad accesso casuale.

51 `Errore interno`

Si è verificato un errore interno in GW-BASIC. Rivolgersi al rivenditore ed illustrare le circostanze in cui l'errore si è verificato.

52 `Numero di file errato`

Un'istruzione o comando fa riferimento ad un file con un numero non aperto oppure oltre l'intervallo numerico specificati per i numeri di file in fase di inizializzazione.

53 `File non trovato`

Un'istruzione LOAD, KILL, NAME, FILES o OPEN fa riferimento ad un file che non esiste sul dischetto.

54 `Modalità del file errata`

Si è tentato di usare PUT, GET o LOF con un file sequenziale o LOAD con un file casuale. Oppure si è tentato di eseguire un OPEN con una modalità di file diversa da I, O, A o R.

55 `File già aperto`

Un OPEN in modalità output sequenziale è emesso per un file già aperto, oppure è stato usato KILL con un file aperto.

56 `Errore non riproducibile`

Non è disponibile alcun messaggio d'errore per il tipo di errore verificatosi. Ciò è determinato generalmente da un errore con un codice indefinito.

57 `Errore I/O della periferica`

Generalmente un errore I/O di disco, generalizzato per includere tutte le periferiche I/O. Si tratta di un errore fatale; il sistema operativo non può cioè rimediare all'errore.

58 `Il file esiste già`

Il nome del file specificato in un'istruzione NAME è identico ad un nome di file già in uso sul dischetto.

59-60 `Errore non riproducibile`

Non è disponibile alcun messaggio d'errore per il tipo di errore verificatosi. Ciò è determinato generalmente da un errore con un codice indefinito.

61 `Disco pieno`

Tutto lo spazio di memoria di disco è in uso.

62 `INPUT dopo la fine`

Un'istruzione INPUT è eseguita dopo che tutti i dati nel file sono stati inseriti, oppure per un file non valido (vuoto). Per evitare questo errore, usare la funzione EOF per individuare la fine di file.

63 `Numero di record errato`

In un'istruzione PUT o GET, il numero di record è maggiore del massimo permesso (16.777.215) o uguale a zero.

64 `Nome di file errato`

E' stata usata una forma non ammessa per il nome del file con LOAD, SAVE, KILL o OPEN; ad esempio, un nome di file con troppi caratteri.

65 `Errore non riproducibile`

Non è disponibile alcun messaggio d'errore per il tipo di errore verificatosi. Ciò è determinato generalmente da un errore con un codice indefinito.

66 `Istruzione diretta nel file`

Si è incontrata un'istruzione diretta mentre si caricava un file di formato ASCII. Il caricamento è arrestato.

67 `Troppi file`

Si è tentato di creare un nuovo file (usando SAVE o OPEN), ma tutte le voci di directory sono piene o le specificazioni di file non sono valide.

68 `Periferica non disponibile`

Si è tentato di aprire un file ad una periferica non esistente. Può darsi che l'hardware non supporti determinate unità periferiche, come LPT2: o LPT3:, oppure che esse siano state disabilitate dall'utente stesso. Ciò si verifica se viene eseguita un'istruzione OPEN "COM1: ma l'utente disattiva il supporto RS-232 con il parametro /c: nella riga di comando.

69 `Buffer di comunicazione pieno`

Si verifica quando viene eseguita un'istruzione di inserimento di comunicazioni, ma la coda di input è già piena. Usare un'istruzione ON ERROR GOTO per riprovare l'input. Gli input successivi tentano di rimediare all'errore a meno che i caratteri continuino ad essere ricevuti più velocemente di come il programma possa elaborarli.

In questo caso sono disponibili alcune opzioni:

- Aumentare la misura del buffer di ricezione COM con il parametro /c:.
- Effettuare un protocollo di sincronizzazione con il satellite/ospitante (ad esempio: XON/XOFF, come dimostrato nell'esempio di programma TTY) per arrestare la trasmissione per un tempo sufficiente a rimettersi in pari.
- Usare una tasso di baud più basso per trasmissione e ricezione.

70 `Permesso negato`

Si è verificato uno dei tre errori di disco restituiti dal sistema di controllo di dischetto.

- Si è tentato di scrivere in un dischetto protetto dalla scrittura.
- Si è tentato di avere accesso ad un file già in uso.
- L'intervallo specificato con UNLOCK non corrisponde all'istruzione LOCK precedente.

71 `Disco non pronto`

Si verifica quando la porta dell'unità per dischi flessibili è aperta o non vi è alcun dischetto nell'unità. Usare un'istruzione ON ERROR GOTO per il recupero.

72 `Errore del disco`

Avviene quando viene scoperto un difetto di hardware. Indica generalmente supporti (media) danneggiati. Copiare tutti i file esistenti su un nuovo dischetto e riformattare il dischetto difettoso. FORMAT fornisce una lista dei canali guasti nella FAT. La parte rimanente del dischetto è nuovamente utilizzabile.

73 `Funzione non ancora predisposta`

Si è tentato di usare una parola riservata che non è disponibile in questa versione di GW-BASIC.

74 `Tentativo di rinominare su altro disco`

Si verifica quando si tenta di rinominare un file con un nuovo nome su un disco diverso da quello relativo su cui si trova il file con il vecchio nome. L'operazione non è eseguita.

75 `Errore di accesso al percorso/file`

In un'operazione OPEN, MKDIR, CHDIR o RMDIR, MS-DOS non è in grado di fare un corretto collegamento percorso di ricerca-nome del file. L'operazione non è completata.

76 `Percorso non trovato`

In un'operazione OPEN, MKDIR, CHDIR o RMDIR, MS-DOS non è in grado di trovare il percorso di ricerca specificato. L'operazione non è completata.

Appendice B
Le funzioni matematiche

Le funzioni matematiche non contenute in GW-BASIC possono essere calcolate nel seguente modo:

Funzione	*Equivalente in GW-BASIC*
Secante	SEC(X)=1/COS(X)
Cosecante	CSC(X)=1/SIN(X)
Cotangente	COT(X)=1/TAN(X)
Seno inverso	ARCSIN(X)=ANT(X/SQR(-X*X+1))
Coseno inverso	ARCCOS(X)=ATN (X/SQR(-X*X+1))+Π/2
Secante inversa	ARCSEC(X)=ATN(X/SQR(X*X-1))+SGN(SGN(X)-1)*Π/2
Cosecante inversa	ARCCSC(X)=ATN(SQR(X/SQR(X*X-1))+SGN(X)-1)*Π/2
Cotangente inversa	ARCCOT(X)=ATN(X)+Π/2
Seno iperbolico	SINH(X)=(EXP(X)-EXP(-X))/2
Coseno iperbolico	COSH(X)=(EXP(X)+EXP(-X))/2
Tangente iperbolica	TANH(X)=(EXP(X)-EXP(-X))/(EXP(X)+EXP(-X))
Secante iperbolica	SECH(X)=2/(EXP(X)+EXP(-X))
Cosecante iperbolica	CSCH(X)=2/(EXP(X)-EXP(-X))
Cotangente iperbolica	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
Seno iperbolico inverso	ARCSINH(X)=LOG(X/SQR(X*X+1))
Coseno iperbolico inverso	ARCCOSH(X)=LOG(X+SQR(X*X-1))

Tangente iperbolica inversa	ARCTANH(X)=LOG((1+X)/(1-X))/2
Cosecante iperbolica inversa	ARCCSCH(X)=LOG(SGN(X)*SQR(X*X+1)+1)/X
Secante iperbolica inversa	ARCSECH(X)=LOG(SQR(-X*X+1)+1)/X
Cotangente iperbolica inversa	ARCCOTH(X)=LOG((X+1)/(X-1))/2

Appendice C
Codici di carattere ASCII

Dec	*Ott*	*Esa*	*Car*	*Dec*	*Ott*	*Esa*	*Car*
000	000	00H	NUL	032	040	20H	SP
001	001	01H	SOH	033	041	21H	!
002	002	02H	STX	034	042	22H	"
003	003	03H	ETX	035	043	23H	#
004	004	04H	EOT	036	044	24H	$
005	005	05H	ENQ	037	045	25H	%
006	006	06H	ACK	038	046	26H	&
007	007	07H	BEL	039	047	27H	'
008	010	08H	BS	040	050	28H	(
009	011	09H	HT	041	051	29H	)
010	012	0AH	LF	042	052	2AH	*
011	013	0BH	VT	043	053	2BH	+
012	014	0CH	FF	044	054	2CH	,
013	015	0DH	CR	045	055	2DH	-
014	016	0EH	SO	046	056	2EH	.
015	017	0FH	SI	047	057	2FH	/
016	020	10H	DLE	048	060	30H	0
017	021	11H	DC1	049	061	31H	1
018	022	12H	DC2	050	062	32H	2
019	023	13H	DC3	051	063	33H	3
020	024	14H	DC4	052	064	34H	4
021	025	15H	NAK	053	065	35H	5
022	026	16H	SYN	054	066	36H	6
023	027	17H	ETB	055	067	37H	7
024	030	18H	CAN	056	070	38H	8
025	031	19H	EM	057	071	39H	9
026	032	1AH	SUB	058	072	3AH	:
027	033	1BH	ESC	059	073	3BH	;
028	034	1CH	FS	060	074	3CH	<
029	035	1DH	GS	061	075	3DH	=
030	036	1EH	RS	062	076	3EH	>
031	037	1FH	US	063	077	3FH	?

Dec=Decimale, Ott=Ottale, Esa=Esadecimale(H), Car=Carattere, LF=Line feed (avanzamento di riga), FF=Form feed, CR=Carriage return (ritorno di riga), DEL=Rubout

Appendice C (*continuazione*)

Dec	*Ott*	*Esa*	*Car*
064	100	40H	@
065	101	41H	A
066	102	42H	B
067	103	43H	C
068	104	44H	D
069	105	45H	E
070	106	46H	F
071	107	47H	G
072	110	48H	H
073	111	49H	I
074	112	4AH	J
075	113	4BH	K
076	114	4CH	L
077	115	4DH	M
078	116	4EH	N
079	117	4FH	O
080	120	50H	P
081	121	51H	Q
082	122	52H	R
083	123	53H	S
084	124	54H	T
085	125	55H	U
086	126	56H	V
087	127	57H	W
088	130	58H	X
089	131	59H	Y
090	132	5AH	Z
091	133	5BH	[
092	134	5CH	\
093	135	5DH	]
094	136	5EH	^
095	137	5FH	_
096	140	60H	`
097	141	61H	a
098	142	62H	b
099	143	63H	c
100	144	64H	d
101	145	65H	e
102	146	66H	f
103	147	67H	g
104	150	68H	h
105	151	69H	i
106	152	6AH	j
107	153	6BH	k
108	154	6CH	l
109	155	6DH	m
110	156	6EH	n
111	157	6FH	o
112	160	70H	p
113	161	71H	q
114	162	72H	r
115	163	73H	s
116	164	74H	t
117	165	75H	u
118	166	76H	v
119	167	77H	w
120	170	78H	x
121	171	79H	y
122	172	7AH	z
123	173	7BH	{
124	174	7CH	|
125	175	7DH	}
126	176	7EH	~
127	177	7FH	DEL

Dec=Decimale, Ott=Ottale, Esa=Esadecimale(H), Car=Carattere, LF=Line feed (avanzamento di riga), FF=Form feed, CR=Carriage return (ritorno di riga), DEL=Rubout

Appendice D
Le subroutine in linguaggio di assemblaggio

Questa appendice è stata creata principalmente per gli utenti esperti nella programmazione in linguaggio di assemblaggio.

Attraverso la funzione USR e l'istruzione CALL, GW-BASIC permette all'utente l'interfaccia con le routine in linguaggio di assemblaggio.

La funzione USR permette il richiamo delle subroutine in linguaggio di assemblaggio in modo simile a come vengono chiamate le funzioni contenute in GW-BASIC. Comunque, per l'interfaccia tra i programmi in linguaggio macchina e GW-BASIC, l'istruzione CALL è da considerarsi più adatta. L'istruzione CALL, a differenza di USR, è compatibile con più linguaggi, crea codici d'origine più leggibili, e può passare argomenti multipli.

D.1 Assegnazione di memoria

Prima di poter caricare una subroutine in linguaggio di assemblaggio, bisogna riservare dello spazio in memoria. Per eseguire ciò, si consigliano i tre modi sotto elencati:

- Specificare una matrice ed usare VARPTR per individuarne l'inizio prima di ogni accesso.
- Nella riga di comando, usare il parametro **/m**. Ottenere il segmento dati di GW-BASIC (DS) e sommare la dimensione del DS per creare un riferimento allo spazio riservato sopra al segmento dati.

- Eseguire un file .COM che rimane residente e memorizzare in esso un puntatore in un settore inutilizzato del vettore di interruzione.

Per caricare le routine in linguaggio di assemblaggio vi sono tre diversi modi:

- Caricare il file con BLOAD. Usare l'istruzione DEBUG per caricare un file .EXE che si trova in memoria alta; eseguire GW-BASIC ed usare BSAVE per salvare il file .EXE.
- Eseguire un file .COM contenente le routine. Salvare il puntatore di queste routine in punti del vettore di interruzione inutilizzati, in modo da permettere al programma in GW-BASIC di ottenere il puntatore e quindi di usare le routine.
- Porre le routine nell'area specificata.

D.2 L'istruzione CALL

CALL *nomevariabile*[(*argomenti*)]

nomevariabile contiene l'offset del segmento corrente della routine in richiamo.

Gli *argomenti* rappresentano le variabili o le costanti da passare alla routine, separate da virgole.

Per ciascun parametro contenuto in *argomenti*, l'offset di 2 byte della posizione del parametro all'interno del segmento dati (DS) viene posto sullo stack.

Il segmento del codice di indirizzo di ritorno di GW-BASIC (CS) e l'offset (IP) vengono spinti sullo stack.

Una chiamata lunga all'indirizzo del segmento fornito nell'ultima istruzione DEF SEG e l'offset fornito in *nomevariabile* trasferiscono il controllo alla routine dell'utente.

Il segmento di stack (SS), il segmento dati (DS), i segmenti "extra" (ES) ed il puntatore dello stack (SP) devono essere conservati.

La figura D.1 mostra lo stato dello stack al momento dell'istruzione CALL:
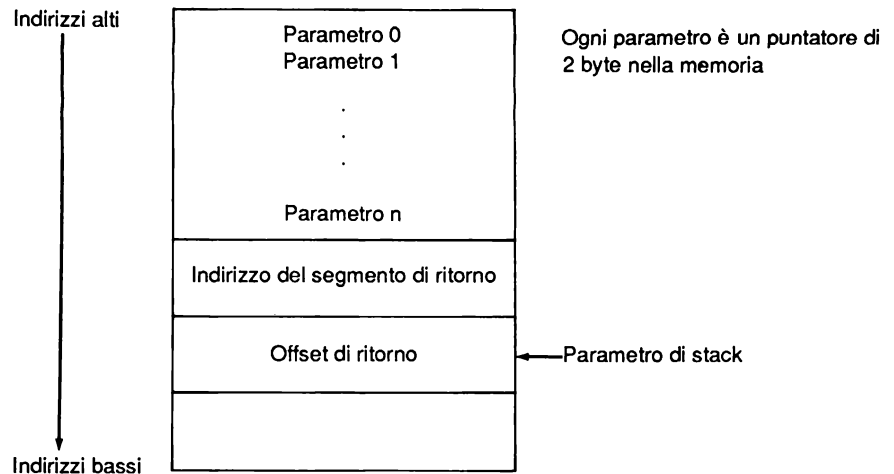
Figura D.1 Aspetto dello stack quando viene attivata l'istruzione CALL

A questo punto il controllo risiede nella routine dell'utente. Si può fare riferimento ai parametri spostando il puntatore di stack (SP) al puntatore di base (BP) ed aggiungendo un offset positivo al puntatore di base.

All'inserimento, i registri di segmento DS, ES e SS puntano tutti al segmento contenente il codice dell'interprete di GW-BASIC. Il registro del segmento del codice CS contiene l'ultimo valore fornito da DEF SEG. Se nessun DEF SEG è stato specificato, il registro punta allo stesso indirizzo al quale puntano DS, ES e SS (il valore predefinito di DEF SEG).

La figura D.2 mostra le condizioni dello stack durante l'esecuzione della subroutine chiamata:
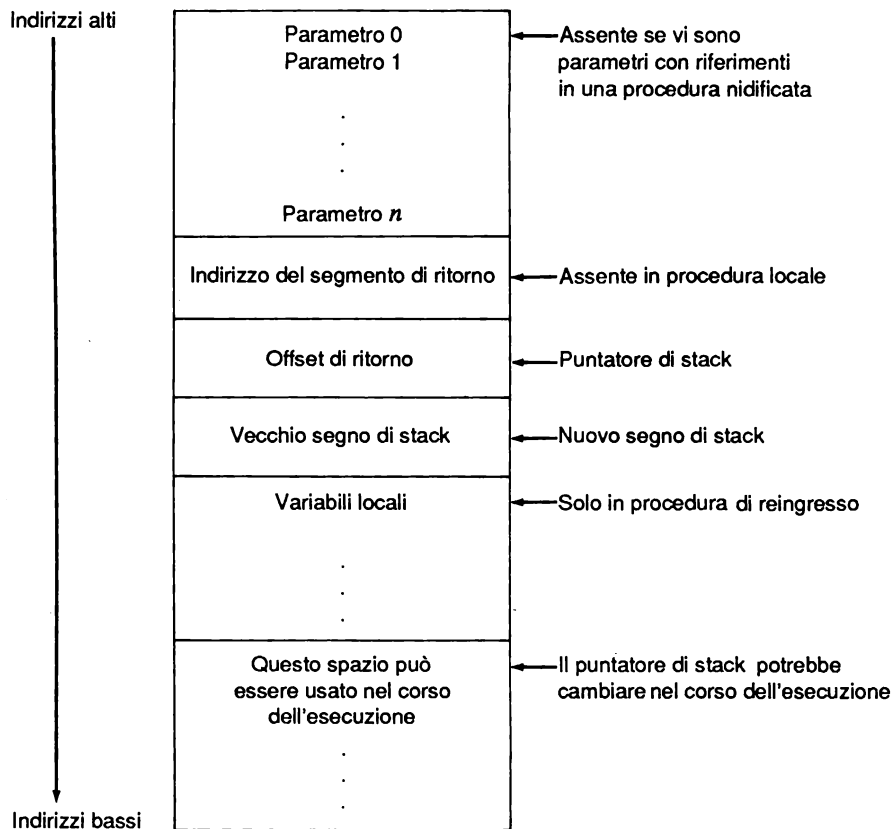
Figura D.2 Aspetto dello stack durante l'esecuzione di un'istruzione CALL

Durante la codificazione di una subroutine, bisogna osservare le sette regole seguenti:

1. La subroutine chiamata potrebbe distruggere i contenuti dei registri AX, BX, CX, DX, SI, DI e BP. Ritornando al programma di GW-BASIC, tali registri non hanno bisogno di essere ripristinati. E' però necessario ripristinare i registri di segmento e il puntatore di stack. Le interruzioni attivate e disattivate andrebbero sempre ripristinate allo stato in cui erano state inserite.

2. Il programma chiamato deve conoscere il numero e la lunghezza dei parametri passati. I riferimenti ai parametri sono offset positivi aggiunti a BP, nell'ipotesi che la routine chiamata abbia spostato il puntatore di stack corrente in BP; cioè, MOV BP,SP. Quando si passano tre parametri, la posizione di PO è a BP+10, quella di P1 a BP+8 e quella di P2 a BP+6.
3. Per porre lo stack alla posizione iniziale della sequenza di chiamata, la routine deve eseguire un RETURN *n (n* è due volte il numero dei parametri nell'elenco degli argomenti). Inoltre, i programmi devono essere definiti con l'istruzione PROC FAR.
4. I valori vengono restituiti a GW-BASIC mediante l'inclusione del nome della variabile che riceve il risultato.
5. Se l'argomento è una stringa, l'offset punta a tre byte chiamati *descrittore di stringa*. Il byte 0 del descrittore contiene la lunghezza della stringa (da 0 a 255). I byte 1 e 2, rispettivamente, rappresentano gli otto bit inferiori e gli otto bit superiori dell'indirizzo di partenza della stringa nello spazio di stringa.

   Avvertenza La routine chiamata non deve cambiare i contenuti di nessuno dei tre byte del descrittore di stringa.

6. Le routine dell'utente possono alterare le stringhe, ma non ne possono modificare la lunghezza. La modifica della lunghezza delle stringhe impedisce a GW-BASIC di gestirle correttamente.
7. Se l'argomento è dato da caratteri letterali nel programma, il descrittore di stringa punta al testo del programma. Fare attenzione a non distruggere o modificare il programma in questo modo. Per evitare risultati imprevedibili, aggiungere +"" dopo i caratteri nel programma. Ad esempio, la riga seguente impone la copia dei caratteri letterali nello spazio di stringa assegnato fuori dallo spazio di memoria del programma:

```
20 A$="BASIC"+""
```

La stringa può poi essere modificata senza influenzare il programma.

Esempi

```
100 DEF SEG=&H2000
110 ACC=&H7FA
120 CALL ACC(A,B$,C)
.
.
.
```

Nella riga 100 il segmento viene impostato a 2000 esadecimale. Il valore della variabile ACC viene aggiunto all'indirizzo mentre la parola bassa dopo il valore di DEF SEG è spostata a sinistra di quattro bit (questa è una funzione del microprocessore, non di GW-BASIC). ACC viene impostato a &H7FA, in modo che al suo richiamo venga eseguita la subroutine in posizione 2000:7FA esadecimale.

All'inserimento, rimangono disponibili soltanto 16 byte (otto parole) all'interno dello spazio di stack assegnato. Se il programma chiamato richiede ulteriore spazio di stack, il programma dell'utente deve reimpostare il puntatore dello stack ad un nuovo spazio assegnato. Nel ritornare al programma di GW-BASIC, assicurarsi di reimpostare la posizione del puntatore, precedentemente modificata, all'inizio della sequenza di chiamata.

L'istruzione in linguaggio di assemblaggio che segue, mostra l'accesso ai parametri passati e la memorizzazione di un risultato di ritorno nella variabile C.

Avvertenza Il programma chiamato deve conoscere il tipo di variabile dei parametri numerici passati. Nei prossimi esempi, l'istruzione seguente copia soltanto due byte:

```
MOVSW
```

Se le variabili A e C sono intere, questa istruzione è appropriata. Se A e C sono invece a precisione singola o doppia, bisognerebbe copiare rispettivamente quattro ed otto byte.

```
MOV BP,SP        Ottiene la posizione corrente dello
                 stack in BP
MOV BX,8[BP]     Ottiene l'indirizzo della descrizione
                 di B$
MOV CL,[BX]      Ottiene la lunghezza di B$ in CL
MOV DX,1[BX]     Ottiene l'indirizzo del descrittore di
                 stringa B$ in DX
MOV SI,10[BP]    Ottiene l'indirizzo di A in SI
MOV DI,6[BP]     Ottiene il puntatore di C in DI
MOVSW            Memorizza la variabile A in 'C'
RET 6            Ripristina lo stack; ritorna
```

D.3 Le chiamate di funzione USR

Nonostante la funzione CALL venga considerata più adatta per eseguire le chiamate di routine in linguaggio di assemblaggio, URS è tuttora disponibile per ragioni di compatibilità con i programmi sviluppati in precedenza.

Sintassi

USR[*n*](*argomento*)

n rappresenta un numero da 0 a 9 che specifica la routine USR da richiamare (vedere l'istruzione DEF USR). Se si omette *n*, il valore predefinito è 0.

argomento rappresenta qualsiasi espressione numerica o alfanumerica.

In GW-BASIC, per assicurarsi che il segmento del codice punti alla subroutine da richiamare, l'istruzione DEF SEG dovrebbe essere eseguita prima della funzione USR. L'indirizzo del segmento fornito nell'istruzione DEF SEG determina il segmento di partenza della subroutine.

Per definire l'offset di chiamata della funzione URS, bisogna aver eseguito un'istruzione DEF SEG per ciascuna funzione USR. Questo offset, con l'indirizzo di DEF SEG attivo, determina l'indirizzo di partenza della subroutine.

Quando viene effettuata la chiamata della funzione USR, il registro AL contiene l'indicatore NTF (number type flag), che specifica il tipo di argomento fornito. Il valore di NTF può essere uno dei seguenti:

Valore di NTF	*Specifica*
2	un valore intero di due byte (formato complemento a due)
3	una stringa
4	un numero a precisione singola e punto mobile
8	un numero a precisione doppia e punto mobile

Se l'argomento di una chiamata di funzione USR è un numero (AL<>3), il valore dell'argomento viene posto nell'accumulatore a punto mobile (FAC). L'accumulatore a punto mobile ha una lunghezza di 8 byte e rappresenta il segmento dati di GW-BASIC. Il registro BX punterà al quinto byte dell'accumulatore.

La figura D.3 mostra la rappresentazione di tutti i tipi di numeri di GW-BASIC in FAC:
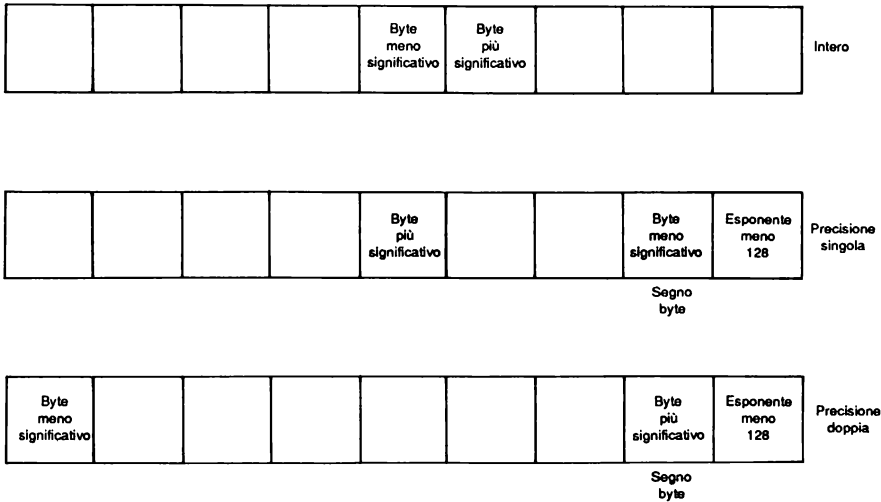
Figura D.3 Tipi di numeri nell'accumulatore a punto mobile

Se l'argomento è un numero a precisione singola e punto mobile:

- L'esponente è BX+3 meno 128. Il punto decimale si trova a sinistra del bit più significativo della mantissa.
- BX+2 contiene i sette bit più in alto nella mantissa con l'1 di testa soppresso (sottointeso). Il bit 7 è il segno del numero (0=positivo, 1=negativo).

- BX+1 contiene gli 8 bit al centro della mantissa.
- BX+0 contiene gli 8 bit più in basso della mantissa.

Se l'argomento è un intero:

- BX+1 contiene gli 8 bit superiori dell'argomento.
- BX+0 contiene gli 8 bit inferiori dell'argomento.

Se l'argomento è un numero a precisione doppia e punto mobile:

- BX+0 a BX+3 sono gli stessi visti per la precisione singola.
- BX-1 a BX-4 contengono 4 byte in più di mantissa. BX-4 contiene gli 8 bit più in basso della mantissa.

Se l'argomento è una stringa (indicata dal valore 3 memorizzato nel registro AL) la combinazione del registro (DX) punta ai tre byte che costituiscono il descrittore di stringa. Il byte 0 del descrittore di stringa contiene la lunghezza della stringa (da 0 a 255). I byte 1 e 2, rispettivamente, rappresentano gli otto bit inferiori e superiori dell'indirizzo di partenza della stringa nel segmento dati di GW-BASIC.

Se l'argomento nel programma è dato da caratteri letterali nel programma, il descrittore di stringa punta al testo del programma. Fare attenzione a non modificare o distruggere il programma in questo modo (vedere quanto detto a proposito dell'istruzione CALL).

Normalmente, il valore restituito dalla chiamata di una funzione USR è dello stesso tipo (intero, di stringa, a precisione singola, o a precisione doppia) dell'argomento passatole. I registri da conservare sono gli stessi dell'istruzione CALL.

Per uscire dalla subroutine USR, bisogna effettuare un ritorno lontano. Il valore restituito deve essere memorizzato nel FAC.

D.4 Programmi che chiamano programmi in linguaggio di assemblaggio

In questa sezione vengono mostrati due esempi dei programmi di GW-BASIC che:

- caricano una routine in linguaggio di assemblaggio per la somma di due numeri
- restituiscono il risultato in memoria
- rimangono residenti in memoria

Il segmento del codice della prima routine viene memorizzato nel vettore di interruzione a 0:100H.

Il primo esempio chiama una subroutine in linguaggio di assemblaggio:

Esempio 1

```
10 DEF SEG=0
100 CS=PEEK(&H102)+PEEK(&H103)*256
200 OFFSET=PEEK(&H100)+PEEK(&H101)*256
250 DEF SEG
300 C1%=2:C2%=3:C3%=0
400 TWOSUM=OFFSET
500 DEF SEG=CS
600 CALL TWOSUM(C1%,C2%,C3%)
700 PRINT C3%
800 END
```

La subroutine in linguaggio di assemblaggio chiamata deve essere assemblata, collegata, e convertita in un file .COM. Se eseguito prima del programma di GW-BASIC, il programma rimane in memoria fino alla disattivazione o al riavviamento del sistema.

```
0100            org 100H
0100            doppio segmento
                assumere cs:doppio
0100 EB 17 9    start:   jmp        start1
0103            usrprg   proc far
0103 55         push bp
```

```
0104 8B EC        mov bp,sp
0106 8B 76 08     mov si, [bp]+8        ;ottenere
                                        ;l'indirizzo del
                                        ;parametro b
0109 8B 04        mov ax,[si]           ;ottenere il
                                        ;valore di b
010B 8B 76 0A     mov si,[bp]+10        ;ottenere
                                        ;l'indirizzo del
                                        ;parametro a
010E 03 04        add ax,[si]           ;aggiungere il
                                        ;valore di a al
                                        ;valore di b
0110 8B 7E 06     mov di,[bp]+6         ;ottenere
                                        ;l'indirizzo del
                                        ;parametro c
0113 89 05        mov di,ax             ;memorizzare la
                                        ;somma del
                                        ;parametro c
0115 5D           pop dp
0116 ca 0006      ret 6
0119              usrprg endp
                                        ;
                                        ;Programma per
                                        ;l'inserimento di
                                        ;una routine in
                                        ;memoria.
                                        ;L'offset ed il
                                        ;segmento vengono
                                        ;memorizzati in
                                        ;posizione
                                        ;100-103H.
0119              start1:
0119 B8 0000      mov ax,0
011C 8E D8        mov ds,ax             ;segmento dei dati
                                        ;all'indirizzo
                                        ;0000H
011E BB 0100      mov bx,0100H          ;puntatore al
                                        ;vettore dell'int
                                        ;100H
0121 83 7F 02 0 cmp word ptr [bx],0
0125 75 16        jne quit              ;programma già
                                        ;eseguito, esci
```

```
0127 83 3F 00   cmp word ptr2 [bx],0
012A 75 11      jne quit             ;programma già
                                     ;eseguito, esci
012C B8 0103 R  mov ax,cs
012F 89 07      mov [bx],ax          ;offset programma
0131 8C c8      mov ax,cs
0133 89 47 02   mov [bx+2],ax        ;segmento dati
0136 0E         push cs
0137 1F         pop ds
0138 BA 0141 R  mov dx,offset veryend
013B CD 27      int 27h
013D            quit:
013D CD 20      int 20h
013F            veryend:
013F            double ends
                end start
```

L'esempio 2 pone la subroutine in linguaggio di assemblaggio nell'area specificata:

Esempio 2

```
10 I=0:JC=0
100 DIM A%(23)
150 MEM%=VARPTR(A%(1))
200 FOR I=1 TO 23
300 READ JC
400 POKE MEM%,JC
450 MEM%=MEM%+1
500 NEXT
600 C1%=2:C2%=3:C3%=0
700 TWOSUM=VARPTR(A%(1))
800 CALL TWOSUM(C1%,C2%,C3%)
900 PRINT C3%
950 END
1000 DATA &H55,&H8b,&Hec
&H8b,&H76,&H08,&H8b,&H04,&H8b,&H76
1100 DATA &H0a,&H03,&H04,&H8b,&H7e,&H06,&H89,&H05,&H5d
1200 DATA &Hca,&H06,&H00
```

Appendice E
La conversione dei programmi BASIC in GW-BASIC

Prima di essere eseguiti, i programmi scritti in BASIC anziché in GW-BASIC, potrebbero richiedere dei piccoli aggiustamenti.

E.1 Dimensioni delle stringhe

Cancellare tutte le istruzioni usate per l'impostazione della lunghezza delle stringhe. Un'istruzione come la seguente:

```
DIM A$(I,J)
```

che imposta le dimensioni di una matrice di stringa di J elementi e di lunghezza I, dovrebbe essere convertita nell'istruzione seguente:

```
DIM A$(J)
```

Per la concatenazione delle stringhe alcuni programmi BASIC usano una virgola od una "e" commerciale (&). Ciascuno di questi caratteri deve essere cambiato in un segno più (+), che è l'operatore di GW-BASIC per la concatenazione delle stringhe.

In GW-BASIC, le funzioni MID$, RIGHT$, e LEFT$, vengono usate per lo spostamento di sottostringhe. Istruzioni del tipo A$(N) per accedere al n-esimo carattere in A$, oppure A$(I,J) per portare una sottostringa dalla posizione I alla posizione J, devono essere modificate nel seguente modo:

Altro BASIC	*GW-BASIC*
X$=A$(I)	X$=MID$(A$,I,1)
X$=A$(I,J)	X$=MID$(A$,I,J-I+1)

Se il riferimento alla sottostringa si trova sul lato sinistro di un'assegnazione e X$ viene usata per sostituire i caratteri in A$, la conversione è la seguente:

Altro BASIC	*GW-BASIC*
A$(I)=X$	MID$(A$,I,1)=X$
A$(I,J)=X$	MID$(A$,I,J-I+1)=X$

E.2 Assegnazioni multiple

Per impostare il valore di B e di C a zero, alcuni linguaggi BASIC permettono l'uso di istruzioni del tipo seguente:

```
10 LET B=C=0
```

GW-BASIC interpreterebbe il secondo segno di uguaglianza (=) come un operatore logico e, se C avesse il valore 0, B verrebbe impostato al valore -1. Occorre perciò convertire questa istruzione in un'istruzione duplice:

```
10 C=0:B=0
```

E.3 Istruzioni multiple

Per separare istruzioni multiple in una riga, alcuni linguaggi BASIC utilizzano un backslash (\). Con GW-BASIC, invece, gli elementi nella riga vengono separati con i due punti (:).

E.4 Funzioni MAT

Per essere eseguiti correttamente, i programmi che includono funzioni MAT (disponibili in alcuni linguaggi BASIC), devono essere riscritti con cicli FOR-NEXT.

E.5 Cicli FOR NEXT

Alcuni linguaggi BASIC eseguono il ciclo FOR-NEXT almeno una volta, indipendentemente dai limiti impostati. Con GW-BASIC il controllo dei limiti impostati è invece preventivo: se i limiti sono stati superati sin dall'inizio, il ciclo non viene eseguito neppure una volta.

Appendice F
Le comunicazioni

Questa appendice fornisce informazioni riguardanti le istruzioni di GW-BASIC necessarie per supportare le comunicazioni asincrone RS-232 con altri computer e unità periferiche.

F.1 Apertura di file di comunicazioni

L'istruzione OPEN COM assegna un buffer per l'input e l'output nello stesso modo in cui l'istruzione OPEN apre i file di disco.

F.2 Comunicazioni I/O

Visto che la porta di comunicazione viene aperta come un file, tutte le istruzioni di I/O valide per i file di disco, sono valide anche per COM.

Le istruzioni per l'input sequenziale di COM sono identiche a quelle usate per i file di disco:

INPUT#
LINE INPUT#
INPUT$

Le istruzioni per l'output sequenziale di COM sono identiche a quelle usate per i dischetti:

PRINT#
PRINT#USING

Per informazioni più dettagliate riguardo a queste istruzioni, vedere la *Guida di riferimento.*

F.3 Le funzioni I/O di COM

L'aspetto più difficile delle comunicazioni asincrone è l'elaborazione dei caratteri alla stessa velocità in cui vengono ricevuti. Ad una velocità superiore a 2400 baud (bps), bisogna sospendere la trasmissione dei caratteri per permettere all'unità ricevente di mettersi alla pari. Per determinare questa sospensione, bisogna inviare XOFF (CTRL-S) all'elaboratore trasmittente per sospendere la trasmissione e, se l'applicazione lo supporta, XON (CTRL-Q) per riprendere l'esecuzione.

Per aiutare a prevedere quando si sta per verificare un eccesso di dati trasmessi, GW-BASIC dispone di tre funzioni:

LOC(x) Questa funzione restituisce il numero dei caratteri in attesa di lettura presenti nella coda di input. La coda di input può contenere più di 255 caratteri (dipende dal parametro /c:). Nel caso che il numero dei caratteri nella lista ecceda 255, LOC(x) ne restituisce però soltanto 255. Essendo il limite di capacità di una stringa di 255 caratteri, ciò permette di evitare l'esecuzione di un test sulla dimensione della stringa prima della lettura dei dati in essa.

LOF(x) Questa funzione restituisce la quantità di spazio disponibile nella coda di input; cioè

/c:(*dimensione*)*-numero di caratteri nella coda di input*

LOF può essere usato per individuare l'imminente raggiungimento del limite di capacità della coda di input.

EOF(x) Vero (-1) indica che la coda è vuota. Se vi sono invece caratteri in attesa di lettura, viene restituito Falso (0).

F.4 Errori possibili

Se viene restituito 0 (cioè coda di input piena) da parte di LOC(x), il tentativo di lettura determina l'errore `Buffer di comunicazione pieno.`

Se si verifica una delle condizioni che seguono, si manifesta un `Errore I/O della periferica`: errore di eccedenza dati trasmessi (OE), errore di struttura (FE), interruzione (BI). L'errore viene corretto dagli input successivi, ma il carattere che ha provocato l'errore è perso.

La perdita del "Data set ready" (DSR) durante un'operazione di I/O determina l'errore `Errore nella periferica.`

L'uso dell'opzione PE (abilitazione parità) nell'istruzione OPEN COM, seguito dalla restituzione di parità errata, determina un `Errore nella parità della periferica.`

F.5 La funzione INPUT$

Per la lettura di file COM, la funzione INPUT$ viene considerata più adatta delle istruzioni INPUT e LINE INPUT, in quanto nelle comunicazioni tutti i caratteri ASCII possono essere significativi. INPUT è il meno adatto poichè con esso l'input si arresta non appena si incontra una virgola o un RITORNO.

INPUT$ permette l'assegnazione ad una stringa di tutti i caratteri letti.

INPUT$ restituisce x caratteri dal file y. Per cui, le istruzioni seguenti sono le più efficienti per la lettura di un file COM:

```
10 WHILE NOT EOF(1)
20 A$=INPUT$(LOC(1),#1)
30 ...
40 ... Elaborazione dati restituiti da A$ ...
50 ...
60 WEND
```

La traduzione di questo segmento di programma è: se la coda contiene dati, restituisci il numero dei caratteri contenuti e memorizzali in A$. Se nella lista vi sono più di 255 caratteri, per prevenire il superamento dei limiti, ne vengono restituiti soltanto 255 per volta. In questo caso EOF(1) risulta falso, per cui l'input continua fino all'esaurimento dei dati nella lista.

Le istruzioni GET e PUT per i file COM

Funzione

Permettono I/O a lunghezza fissa per COM.

Sintassi

GET *numerofile, nbyte* **PUT** *numerofile, nbyte*

Commenti

numerofile è un'espressione intera che restituisce un numero di file valido.

nbyte rappresenta un'espressione intera che restituisce il numero di byte da trasferire nel o dal buffer del file. *nbyte* non può eccedere il valore impostato dal parametro /s: all'esecuzione di GW-BASIC.

A causa delle ridotte prestazioni delle comunicazioni tramite linea telefonica, è consigliabile non utilizzare GET e PUT in applicazioni di questo tipo.

Esempio

Il programma di esempio TTY che segue è un esercizio in comunicazioni I/O. Esso è stato concepito per permettere l'uso del computer come un terminale convenzionale. Oltre alle comunicazioni "full-duplex", il programma TTY permette il caricamento di dati in un file. Un file può essere perciò trasmesso ad un altro computer.

Oltre alla dimostrazione di elementi delle comunicazioni asincrone, questo programma viene utilizzato per il trasferimento e l'estrazione di programmi e di dati di GW-BASIC al/dal computer.

Avvertenza Questo programma è stato impostato per comunicare con il sistema DEC(r) SYSTEM-20 specialmente con l'uso di XON e XOFF. Per comunicare con altri tipi di hardware potrebbe essere necessario apportare qualche modifica.

F.6 Il programma di esempio TTY

```
10 SCREEN 0,0:WIDTH 80
15 KEY OFF:CLS:CLOSE
20 DEFINT A-Z
25 LOCATE 25,1
30 PRINT STRING$(60," ")
40 FALSE=0:TRUE=NOT FALSE
50 MENU=5 'Valore della chiave del MENU (^E)
60 XOFF$=CHR$(19) :XON$=CHR$(17)
100 LOCATE 25,1:PRINT "Programma TTY asinc";
110 LOCATE 1,1:LINE INPUT "Velocità?";"VELOCITA$
120 COMFIL$="COM1:,+VELOCITA$+",E,7"
130 OPEN COMFIL$ AS #1
140 OPEN "SCRN:"FOR OUTPUT AS #3
200 PAUSE=FALSE
210 A$=INKEY$:IF A$=""THEN 230
220 IF ASC(A$) =MENU THEN 300 ELSE PRINT #1,A$;
230 IF EOF(1) THEN 210
240 IF LOC(1)>128 THEN PAUSE=TRUE:PRINT #1,XOFF$;
250 A$=INPUT$(LOC(1),#1)
260 PRINT #3,A$;:IF LOC(1)>0 THEN 240
270 IF PAUSE THEN PAUSE=FALSE:PRINT #1,XON$;
280 GOTO 210
300 LOCATE 1,1:PRINT STRING$(30,32):LOCATE 1,1
310 LINE INPUT "FILE?";DSKFIL$
400 LOCATE 1,1:PRINT STRING$(30,32) :LOCATE 1,1
410 LINE INPUT"(T)rasmetti o (R)icevi?";TXRX$
420 IF TXRX$="T" THEN OPEN DSKFIL$ FOR INPUT AS
#2:GOTO 1000
430 OPEN DSKFIL$ FOR OUTPUT AS #2
440 PRINT #1,CHR$(13);
500 IF LOC(1) THEN GOSUB 600
510 IF LOC(1)>128 THEN PAUSE=TRUE:PRINT #1,XOFF$;
520 A$=INPUT$(LOC(1),#1)
530 PRINT #2,A$;:IF LOC(1)>0 THEN 510
540 IF PAUSE THEN PAUSE=FALSE:PRINT #1,XON$;
550 GOTO 500
600 FOR I=1 TO 5000
```

```
610 IF NOT EOF(1) THEN I=9999
620 NEXT I
630 IF I>9999 THEN RETURN
640 CLOSE #2;CLS:LOCATE 25,10:PRINT "* Trasmissione
completata *";
650 RETURN 200
1000 WHILE NOT EOF(2)
1010 A$=INPUT$(1,#2)
1020 PRINT #1,A$;
1030 WEND
1040 PRINT #1,CHR$(28);^Z per chiudere il file.
1050 CLOSE #2:CLS:LOCATE 25,10:PRINT "** Caricamento
completo **";
1060 GOTO 200
9999 CLOSE:KEY ON
```

F.7 Note al programma esempio TTY

Avvertenza Asincrono sottintende un carattere I/O piuttosto che una riga o un blocco I/O. Per cui, tutto quanto stampato (su un file COM o sullo schermo) termina con un punto e virgola (;). Ciò ritarda il ritorno di riga che viene normalmente emesso alla fine dell'istruzione PRINT.

Numero di riga	*Commenti*
10	Imposta lo schermo in modalità alfa in bianco e nero e la larghezza a 80 caratteri.
15	Disattiva la visualizzazione dei tasti di funzione, libera lo schermo, e si accerta che tutti i file siano chiusi.
20	Definisce tutte le variabili numeriche in valori interi, principalmente in funzione delle subroutine in 600-620. Qualsiasi programma in cerca dell'ottimizzazione di velocità dovrebbe usare dei contatori interi ovunque possibile nei cicli.
40	Definisce un vero o falso booleano.
50	Definisce il valore ASCII (ASC) del tasto MENU.
60	Definisce i caratteri ASCII XON e XOFF.

100-130 Stampa il nome del programma e chiede il tasso di baud. Apre le comunicazioni al file numero 1, con parità pari e 7 bit di dati.

200-280 Questa porzione di programma esegue comunicazioni I/O "full-duplex" tra lo schermo e l'unità collegata al RS-232 nel seguente modo:

1. Legge un carattere dalla tastiera ad A$. Se nessun carattere è in attesa, INKEY$ restituisce una stringa nulla.
2. Se un carattere della tastiera è in attesa:

   Se il carattere è il tasto MENU, l'operatore è pronto per la trasmissione del file. Legge perciò il nome del file.

   Se il carattere (A$) non è il tasto MENU, controlla scrivendo al file di comunicazioni (PRINT #1...).
3. Se nessun carattere è in attesa, verifica se si stanno ricevendo dei caratteri.
4. Alla riga 230, verifica se nel buffer di COM vi sono dei caratteri in attesa. In caso negativo, ritorna e controlla la tastiera.
5. Alla riga 240, se vi sono più di 128 caratteri in attesa, imposta l'indicatore PAUSE per indicare la sospensione dell'input. Manda XOFF all'unità di trasmissione, in modo da arrestare ulteriori trasmissioni.
6. Alle righe 250 e 260, legge e visualizza sullo schermo i contenuti del buffer di COM (riga 240). Se la ricezione è in ritardo rispetto alla trasmissione, sospende la trasmissione.
7. Riprende la trasmissione inviando un XON soltanto se era stata precedentemente sospesa con XOFF.
8. Ripete l'esecuzione fino alla pressione del tasto MENU.

300-320 Legge il nome del file di disco al quale trasmettere. Apre il file col numero 2.

400-420 Chiede se il file nominato è da trasmettere o da ricevere.

430	Riceve la routine. Invia un RITORNO all'unità di trasmissione per iniziare la ricezione. In questo programma si suppone che l'ultimo comando mandato all'unità di trasmissione servisse per iniziare questo tipo di trasferimento e mancasse soltanto del RITORNO finale. Se l'unità di trasmissione è un sistema DEC, un comando di questo tipo potrebbe essere: COPY TTY:=MANUAL.MEM (Tasto MENU) se fosse premuto il tasto MENU invece di RITORNO.
500	Quando la ricezione dei caratteri è terminata, (LOC(x) restituisce 0) il programma esegue una routine di timeout.
510	Se non vi sono più di 128 caratteri in attesa, segnala una pausa e invia XOFF all'unità di trasmissione.
520-530	Legge tutti i caratteri nella coda di COM (LOC(x)) e li scrive sul dischetto fino a che la ricezione si è posta allo stesso livello della trasmissione.
540-550	Se viene emessa una pausa, riavvia l'unità di trasmissione inviando XON e liberando l'indicatore di pausa. Continua l'operazione fino a che non viene ricevuto un carattere per un determinato periodo di tempo.
600-650	Subroutine di timeout. Il conteggio del ciclo FOR era determinato da sperimentazione. Se entro 17-20 secondi non si ricevono caratteri, si suppone che la trasmissione sia terminata. In caso contrario (riga 610), il programma imposta *n* ad un valore abbondantemente superiore al limite del ciclo FOR in modo da uscire dal ciclo e ritornare alla routine di chiamata. Se la trasmissione non è terminata, chiude il file di disco e riprende normalmente le attività.
1000-1060	Routine di trasmissione. Legge un carattere in A$ usando l'istruzione INPUT$, fino al termine del file di disco. Manda un carattere all'unità COM alla riga 1020. Nel caso in cui l'unità ricevente richieda la chiusura dei file, al termine del file manda un ^Z alla riga 1040. Le righe 1050 e 1060 chiudono il file di disco, stampano un messaggio di chiusura, e ritornano alla modalità di conversazione alla riga 200.
9999	Non eseguito al momento. Come prova, aggiungere alcune righe alla routine che va dalla riga 400 a 420 per uscire dal programma tramite la riga 9999. Questa riga chiude il file COM lasciato aperto e ripristina la visualizzazione dei tasti di funzione.

Appendice G
Equivalenti esadecimali

Tabella G.1 Equivalenti decimali e binari dei valori esadecimali

Valore esadecimale	*Decimale*	*Binario*
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Tabella G.2 Equivalenti decimali dei valori esadecimali

Valore esadecimale	Decimale	Valore esadecimale	Decimale
0	0	80	128
1	1	.	
2	2	.	
3	3	.	
4	4	90	144
5	5	.	
6	6	.	
7	7	.	
8	8	A0	160
9	9	.	
A	10	.	
B	11	.	
C	12	B0	176
D	13	.	
E	14	.	
F	15	.	
10	16	C0	192
11	17	.	
12	18	.	
13	19	.	
14	20	D0	208
15	21	.	
16	22	.	
17	23	.	
18	24	E0	224
19	25	.	
1A	26	.	
1B	27	.	
1C	28	F0	240
1D	29	100	256
1E	30	200	512
1F	31	300	768
20	32	400	1024
.	.	500	1280
.	.	600	1536
.	.	700	1792

Tabella G.2 Equivalenti decimali dei valori esadecimali

Valore esadecimale	***Decimale***	***Valore esadecimale***	***Decimale***
30	48	800	2048
.	.	900	2304
.	.	A00	2560
.	.	B00	2816
40	64	C00	3072
.	.	D00	3328
.	.	E00	3584
.	.	F00	3840
50	80	1000	4096
.	.	2000	8192
.	.	3000	12288
.	.	4000	16384
60	96	5000	20480
.	.	6000	24576
.	.	7000	28672
.	.	8000	32768
70	112	9000	36864
.	.	A000	40960
.	.	B000	45056
.	.	C000	49152
		D000	53248
		E000	57344
		F000	61440

Appendice H
I codici di tasto

Tastiera	*Codice*
ESC	01
1/!	02
2/@	03
3/#	04
4/$	05
5/%	06
6/^	07
7/&	08
8/*	09
9/(	0A
-/_	0C
=/+	0D
BACKSPACE	0E
TABULAZIONE	0F
Q	10
W	11
E	12
R	13
T	14
Y	15
U	16
I	17
O	18
P	19
[/{	1A
]/}	1B
RITORNO	1C

<table>
<tr><td>CTRL</td><td>1D</td></tr>
<tr><td>A</td><td>1E</td></tr>
<tr><td>S</td><td>1F</td></tr>
<tr><td>D</td><td>20</td></tr>
<tr><td>F</td><td>21</td></tr>
<tr><td>G</td><td>22</td></tr>
<tr><td>H</td><td>23</td></tr>
<tr><td>J</td><td>24</td></tr>
<tr><td>K</td><td>25</td></tr>
<tr><td>L</td><td>26</td></tr>
<tr><td>;/:</td><td>27</td></tr>
<tr><td>’/"</td><td>28</td></tr>
<tr><td>‘/~</td><td>29</td></tr>
<tr><td>SHIFT SINISTRO</td><td>2A</td></tr>
<tr><td>/|</td><td>2B</td></tr>
<tr><td>Z</td><td>2C</td></tr>
<tr><td>X</td><td>2D</td></tr>
<tr><td>C</td><td>2E</td></tr>
<tr><td>V</td><td>2F</td></tr>
<tr><td>B</td><td>30</td></tr>
<tr><td>N</td><td>31</td></tr>
<tr><td>M</td><td>32</td></tr>
<tr><td>,/<</td><td>33</td></tr>
<tr><td>//?</td><td>35</td></tr>
<tr><td>SHIFT DESTRO</td><td>36</td></tr>
<tr><td>*/PRTSC</td><td>37</td></tr>
<tr><td>ALT</td><td>38</td></tr>
<tr><td>BARRA SPAZIATRICE</td><td>39</td></tr>
<tr><td>CAPS LOCK</td><td>3A</td></tr>
<tr><td>F1</td><td>3B</td></tr>
<tr><td>F2</td><td>3C</td></tr>
<tr><td>F3</td><td>3D</td></tr>
<tr><td>F4</td><td>3E</td></tr>
<tr><td>F5</td><td>3F</td></tr>
<tr><td>F6</td><td>40</td></tr>
<tr><td>F7</td><td>41</td></tr>
<tr><td>F8</td><td>42</td></tr>
<tr><td>F9</td><td>43</td></tr>
<tr><td>F10</td><td>44</td></tr>
<tr><td>NUM LOCK</td><td>45</td></tr>
<tr><td>SCROLL LOCK</td><td>46</td></tr>
<tr><td>7/HOME</td><td>47</td></tr>
</table>

8/SU	48
9/PGUP	49
-	4A
4/A SINISTRA	4B
5	4C
6/A DESTRA	4D
+	4E
1/END	4F
2/GIU'	50
3/PGDN	51
0/INS	52
./DEL	53

Appendice I
Caratteri riconosciuti da GW-BASIC

Il set dei caratteri di GW-BASIC comprende tutti i caratteri considerati leciti in comandi, istruzioni, funzioni e variabili di GW-BASIC. L'insieme di questi caratteri può essere alfabetico, numerico o composto da caratteri speciali.

I caratteri alfabetici di GW-BASIC sono le lettere dell'alfabeto minuscole e maiuscole.

I caratteri numerici di GW-BASIC sono le cifre da 0 a 9.

I caratteri che seguono sono caratteri speciali riconosciuti da GW-BASIC:

Carattere	*Descrizione*
	Spazio
=	Segno di uguaglianza o di assegnazione
+	Segno più o di concatenazione di stringhe
-	Segno meno
*	Asterisco o segno di moltiplicazione
/	Slash o segno di divisione
^	Accento circonflesso, segno esponenziale, simbolo del tasto CTRL
(	Parentesi aperta
)	Parentesi chiusa
%	Simbolo di percentuale o di dichiarazione di intero
#	Segno di dichiarazione di valore a precisione doppia
$	Simbolo del dollaro o di dichiarazione di stringa

!	Punto esclamativo o dichiarazione di valore a precisione singola.
[	Parentesi quadra aperta.
]	Parentesi quadra chiusa.
,	Virgola.
""	Virgolette o delimitatori di stringa.
'	Apostrofo, indicatore di commenti.
.	Punto e punto decimale.
;	Punto e virgola o soppressore ritorno di riga (carriage return).
:	Due punti o delimitatore dell'istruzione.
&	"e" commerciale o descrittore conversione numeri esadecimali ed ottali.
?	Punto interrogativo.
<	Segno di minoranza.
>	Segno di maggioranza.
\	Backslash o segno per divisione di interi.
@	Segno "Al".
_	Carattere di sottolineatura.
BACKSPACE	Annulla l'ultimo carattere digitato.
ESC	Cancella la riga logica corrente dallo schermo.
TABULAZIONE	Sposta la posizione di stampa alla posizione di tabulazione che segue. Le posizioni di tabulazione sono impostate ogni otto colonne.
CURSORE	Sposta il cursore alla riga che segue.
RITORNO	Termina l'input in una riga, sposta il cursore all'inizio della riga seguente o esegue l'istruzione (in modalità diretta).

Glossario

accesso

L'operazione di ricerca, lettura o scrittura di dati su un'unità di memoria.

accesso casuale (memoria)

L'area di lavoro ad alta velocità che permette l'accesso alle varie posizioni in memoria utilizzando un sistema a coordinate verticali ed orizzontali. Il computer può scrivere e leggere informazioni nella memoria ad accesso casuale. La memoria ad accesso casuale è spesso detta RAM.

acronimo

Una parola formata dalle lettere iniziali (o da segmenti) di più parole. Ad esempio, COBOL è l'acronimo di COmmon Business Oriented Language.

albero (struttura directory)

Una struttura di organizzazione dei file, composta da un sistema di directory e subdirectory a diversi livelli.

alfabetici (caratteri)

Rappresentazione di dati attraversi caratteri alfabetici piuttosto che con numeri; le lettere dell'alfabeto.

alfanumerici (caratteri)

Una combinazione delle parole alfabetici e numerici; un insieme di caratteri comprendente lettere, numeri e simboli speciali.

algoritmo

Un insieme di regole o procedure ben definite da seguire in modo da ottenere la soluzione di un problema in un numero finito di operazioni. Un algoritmo può richiedere tipi di procedure o di istruzioni aritmetiche, algebriche, logiche ecc.. Un algoritmo può essere più o meno complesso. Tutti gli algoritmi devono però portare ad una soluzione in un numero finito di operazioni.

Per risolvere problemi tramite l'uso del computer, gli algoritmi sono fondamentali, in quanto il computer richiede determinati set di istruzioni in grado di fornire soluzioni in spazi di tempo accettabili.

argomento

1. Un tipo di variabile il cui valore non è una funzione diretta di un'altra variabile. Può rappresentare la posizione di un numero in un'operazione matematica, oppure il numero utilizzato da una funzione per determinare i risultati.
2. Un fattore di riferimento conosciuto necessario per localizzare un elemento (funzione) in una tabella. Ad esempio, nella funzione della radice quadrata SQRT(X), X rappresenta l'argomento. Il valore di X determina il valore restituito da questa funzione.

ASCII

E' l'acronimo di American Standard Code for Information Interchange. ASCII è un codice di 8 bit standardizzato, usato per l'interfaccia dalla maggior parte dei computer.

ASCII, sviluppato dall'American National Standards Institute (ANSI), utilizza 7 bit binari per informazioni e l'ottavo bit come parità.

asincrona (comunicazione)

Un modo per trasmettere i dati da un'unità ad un'altra in ordine sequenziale. Ciascun carattere trasmesso viene preceduto da un bit di partenza e seguito da un bit di stop. Questa operazione viene anche chiamata trasmissione di inizio/fine.

asincrono

1. L'essere fuori sincronia. (*Vedere* Sincrono)
2. Un tipo di operazione di computer nella quale l'inizio di una nuova istruzione avviene al completamento di quella precedente. Per cui, in una sequenza di istruzioni, non è impostata alcuna struttura temporale. Prima di poter iniziare una nuova istruzione, l'esecuzione di quella corrente deve essere stata completata, indipendentemente dalla durata.

assemblaggio (linguaggio di)

Un linguaggio simbolico orientato alla macchina più che alla risoluzione di problemi. Un programma scritto in linguaggio di assemblaggio viene convertito in un programma in linguaggio macchina tramite un assemblatore. I simboli rappresentanti le posizioni in memoria vengono convertiti in posizioni di memoria numeriche; i codici delle operazioni simboliche vengono convertiti in codici di operazioni numeriche.

assemblatore

Un programma che produce un programma in linguaggio macchina eseguibile direttamente dal computer.

avviamento

Una procedura di macchina che permette al sistema di iniziare le operazioni. Le prime istruzioni vengono caricate nel computer dall'unità di input. Queste istruzioni permettono il caricamento della parte rimanente del sistema.

backup

1. Una copia di riserva di dati in un dischetto o in qualche altro dispositivo per la memorizzazione dei dati. Questa precauzione assicura il ripristino dopo l'eventuale perdita o distruzione del supporto di memorizzazione.
2. La disponibilità dell'hardware in sede o in luogo lontano per completare un'operazione in caso di mancato funzionamento dell'hardware principale.

BASIC

E' l'acrinomo di Beginner's All-purpose Symbolic Instruction Code. BASIC è un linguaggio per la programmazione del computer sviluppato nel College Darmouth come strumento istruttivo per l'insegnamento dei concetti di programmazione fondamentali. Da quando è stato creato, BASIC ha acquisito una larga utenza ed è attualmente considerato uno dei linguaggi più facili da imparare.

batch (procedura)

Un metodo di operazione del computer attraverso il quale l'esecuzione di un programma o di un insieme di programmi collegati viene completata prima che un altro tipo di programma possa avere inizio.

baud

Un'unità di misura della velocità di trasmissione dei dati. La velocità in baud è il numero degli elementi trasmessi per secondo. Poichè un elemento può essere costituito da più di un bit, il baud non è sinonimo di bit per secondo. I valori di baud tipici sono 110, 300, 1200, 2400, 4800, e 9600.

binaria (cifra)

Una quantità espressa nelle cifre binarie 0 ed 1.

binario

1. Una caratteristica o una proprietà che implica una scelta o una condizione nella quale vi sono due possibili scelte.
2. Un sistema di numerazione che usa 2 come base anziché 10 (sistema decimale). Nella sua forma scritta, il sistema binario utilizza due sole cifre, 0 ed 1.
3. Un dispositivo che usa soltanto due stati possibili o livelli per eseguire le proprie funzioni. Un computer esegue programmi in forma binaria.

bit

Una contrazione della parole "binary digit" (cifra binaria). Un bit può essere 0 o 1 e rappresenta l'unità d'informazione più piccola riconoscibile dal computer.

blocco

Un ammontare di spazio di memoria, di lunghezza arbitraria, normalmente contiguo e di solito composto da più record collegati tra di loro e trattati come un insieme.

booleana (logica)

Un campo di analisi matematiche nel quale si eseguono confronti. Un'istruzione può effettuare il confronto di due campi di dati e, sulla base dell'esito di questo confronto, può modificare uno dei due campi o un terzo campo. Questo sistema fu formulato dal matematico Britannico George Boole (1815-1864). Alcuni operatori booleani sono OR, AND, NOT, XOR, EQV e IMP.

bps

Bit per secondo.

buffer

Un'area utilizzata per la memorizzazione temporanea, dalla quale i dati vengono poi trasferiti alle varie unità.

byte

Un'unità di dati, composta da otto bit di dati ed un bit di parità, che rappresenta un carattere alfabetico o speciale, due cifre, o otto bit binari. Inoltre, byte viene usato per il riferimento ad una sequenza di otto cifre binarie utilizzate come se facessero parte di un insieme. Byte viene normalmente codificato nel formato ASCII.

calcolo

Una serie di numeri e segni matematici che, all'inserimento nel computer, vengono elaborati in base alle istruzioni specificate.

campo

Una parte di un record assegnata per una determinata categoria di dati.

cancellare

Rimuovere o sostituire dei punti magnetizzati dal mezzo di memorizzazione.

carattere

Qualsiasi lettera dell'alfabeto, numero, segno di punteggiatura e qualsiasi altro simbolo riconosciuto dal computer. Il termine carattere è sinonimo di byte.

ciclo chiuso

Una serie di istruzioni eseguite ripetutamente fino al verificarsi di un determinato evento. La capacità di ricollegarsi a determinate istruzioni per riutilizzarle, elimina la necessità di ripeterne l'inserimento nel programma, ed è una delle caratteristiche più importanti dei programmi memorizzati.

circuito integrato

Un circuito elettronico completo posto in un piccolo semiconduttore.

COBOL

Acronimo per COmmon Business-Oriented Language. Si tratta di un linguaggio di computer adatto per lo sviluppo di programmi applicativi complicati, usati per i problemi di natura economica. COBOL è stato creato da CODASYL, un comitato che rappresenta il Dipartimento della Difesa degli Stati Uniti, alcuni produttori di computer, e grossi utenti di sistemi di data processing. COBOL è stato concepito per dare alla codificazione della gestione di dati e dell'esecuzione di problemi una forma Inglese discorsiva.

codificare

1. Scrivere istruzioni per un sistema di computer
2. Classificare i dati in base a tabelle arbitrarie
3. Usare un linguaggio macchina
4. Programmare

comando

Una pulsazione, segnale o serie di lettere che comunicano al computer di iniziare, terminare o continuare un'operazione. Spesso la parola comando viene erroneamente usata come sinonimo di istruzione.

compatibile

Una descrizione di dati, programmi o hardware, che possono essere utilizzati con più tipi di computer.

compilatore

Un programma di computer che traduce un programma per la gestione di problemi in un programma di istruzioni simili al linguaggio del computer.

concatenazione

1. L'uso di un puntatore in un record per indicare l'indirizzo di un altro record logicamente correlato al primo.
2. L'unificazione dei set di dati, come i file, in modo da formare un set di dati (un file nuovo). Un set di dati concatenati è un insieme di set logicamente collegati tra loro.

configurazione

In hardware, è un insieme di unità collegate tra loro per costituire un sistema. In software, è il totale delle routine e le relative correlazioni.

copia su carta

Una copia stampata, in forma leggibile, dell'output del computer. Ad esempio, un rapporto o dei grafici.

coprocessore

Un'unità di microprocessore collegata al microprocessore centrale, che esegue dei calcoli speciali (del tipo aritmetica a punto mobile) in modo più efficace del processore centrale (CPU) da solo.

costante

Un valore o dato che non cambia mai.

CPU

Vedere Processore centrale

cursore

Una riga o un rettangolo intermittente visualizzato sullo schermo, che indica la posizione per l'inserimento di dati.

dati

Un termine generale usato per riferirsi alle unità di informazione che possono essere prodotte o elaborate da un computer. *Vedere anche* Informazioni.

dato (unità)

La più piccola unità informativa fisica nominata.

debugging

Il processo di controllo della logica di un programma al fine di isolare e eliminare errori.

delimitatore

Un carattere usato per segnare l'inizio e la fine di un'unità di dati in un dispositivo di memorizzazione. I delimitatori usati per separare ed organizzare dati sono i seguenti: virgole, punti e virgola, punti e spazi.

densità semplice

La densità di registrazione standard di un dischetto. I dischetti a densità singola hanno una capacità di memorizzazione di approssimativamente 3400 bit per pollice.

directory

Un elenco che indica il nome, la posizione, la dimensione, e la data della creazione o dell'ultima modifica di ciascun file su un supporto di memorizzazione (ad esempio un dischetto).

dischetto

Un disco flessibile ricoperto da materiale magnetico, racchiuso in un involucro protettivo, usato per la memorizzazione di software e di dati.

disco rigido

Un disco fisso racchiuso in un involucro permanentemente sigillato, che lo protegge dai fattori ambientali. Usato per la memorizzazione di dati.

doppia densità

Un tipo di dischetto con capacità di memorizzazione due volte superiore a quella di un dischetto a densità semplice.

doppia faccia

Si riferisce ad un dischetto che può contenere dati su entrambi i lati.

DOS

L'acronimo per Disk Operating System. E' un insieme di procedure tecniche che abilitano il computer ad operare con un sistema di unità per l'inserimento e la memorizzazione dei dati.

esadecimale

Un sistema di numerazione a base, o radice, 16. I simboli usati in questo sistema sono le cifre decimali da 0 a 9 e sei altre cifre rappresentate con A, B, C, D, E, F.

esponente

Un simbolo che, scritto sulla parte superiore destra di una base, specifica quante volte ripeterne la moltiplicazione. Nell'esempio di A^2, A è la base e 2 è l'esponente. A^2 significa A per A (AxA).

estensione

Un insieme da 1 a 3 caratteri che segue il nome del file. L'estensione, che viene separata dal nome tramite un punto (.), definisce o chiarisce il nome del file.

etichetta di volume

Il nome usato per i contenuti di un dischetto o di una partizione su un disco fisso.

file

Un insieme di dati collegati o di programmi trattato dal computer come un'unità.

file di dati

Un insieme di record di dati collegati tra loro e organizzati in un modo specifico. I file di dati contengono record che, anziché contenere un programma per la gestione dei dati, contengono semplicemente informazioni.

fine anormale dell'operazione

Il termine dell'esecuzione non completa di un'operazione o di un programma causata da un errore non risolvibile dalle routine programmate per la correzione di errori.

fine file (simbolo di) (EOF)

Un simbolo o un equivalente di macchina che indica la lettura dell'ultimo record nel file.

formato

Un'impostazione dei dati predeterminata che struttura la memorizzazione di informazioni su un'unità di memorizzazione esterna.

funzione

Un'azione del computer, come definita da una determinata istruzione. Alcune funzioni di GW-BASIC sono COS, EOF, INSTR, LEFT$, e TAN.

funzione (tasti di)

Tasti la cui pressione produce una particolare istruzione che viene poi eseguita dal computer. La funzione del tasto viene determinata dai programmi applicativi in uso.

GIGO

Acronimo per Garbage In Garbage Out. Ciò significa che, se i dati di input sono errati (garbage in), anche l'output relativo sarà errato (garbage out).

grafica

La capacità dell'hardware/software di visualizzare gli oggetti in forma figurativa, anzichè in parole. Normalmente, questa visualizzazione avviene su un terminale per la visualizzazione grafica con capacità di tracciare righe e che permette l'interazione (ad esempio con una penna luminosa).

hardware

L'impianto fisico che forma un sistema.

I/O

Acronimo di input/output.

indirizzo

Un nome, un'etichetta o un numero che identifica un registro, una posizione o unità in cui vengono memorizzate informazioni.

informazioni

Fatti e conoscenze derivati da dati. Il computer opera su e genera dati. Quanto derivato dai dati è detto appunto informazione. Le due parole non sono sinonimi, anche se sono spesso usate in modo intercambiabile.

input

1. L'esecuzione dell'inserimento dei dati nel computer.
2. I dati stessi inseriti nel computer.

input/output

Un termine generale che si riferisce alle unità comunicanti con il computer. Spesso input/output viene abbreviato I/O.

interfaccia

Un percorso per lo scambio di informazioni, che permette la comunicazione e l'interazione tra parti di un computer, più computer e unità esterne.

intero

Un'entità completa, priva di parti frazionali. L'intero numero o numero naturale. Ad esempio, 65 rappresenta un intero; mentre 65.1 rappresenta un valore con parti frazionali.

interprete

Un programma che legge, traduce ed esegue un programma utente una riga per volta. A differenza dall'interprete, un compilatore, legge e traduce il programma prima di eseguirlo.

istruzione

Un'istruzione in linguaggio di livello superiore che impone al computer l'esecuzione di una sequenza di operazioni in ordine sequenziale.

K

Il simbolo rappresentante la quantità 2^{10} (che è uguale a 1024).

linguaggio algebrico

Un linguaggio le cui istruzioni sono strutturate in modo tale da apparire come espressioni algebriche. Un buon esempio di un linguaggio di questo tipo è Fortran.

logaritmo

Un logaritmo di un numero è il valore dell'esponente a cui bisogna elevare una costante predeterminata, detta base, per ottenere tale numero.

loop

Vedere Ciclo chiuso

M

Simbolo rappresentante la quantità 1,000,000 (10^6). Quando si riferisce alla memoria, questo simbolo indica più precisamente 1,048,576 (2^{20}).

mantissa

La parte frazionale o decimale del logaritmo di un numero. Ad esempio, il logaritmo di 163 è 2.212. La mantissa è 0.212, mentre 2 è la caratteristica.

Quando si riferisce ai numeri a punto mobile, la mantissa rappresenta la parte numerica. Ad esempio, 24 può essere scritto come 24.2, dove 24 è la mantissa e 2 è l'esponente. Il numero a punto mobile viene letto come .24 X 10^2, o 2^4.

matrice

1. Un gruppo organizzato di dati in cui l'argomento viene posto davanti alla funzione.
2. Un insieme di elementi in cui la posizione di ciascun argomento è significativa. Un buon esempio di una matrice è una tabella di moltiplicazione.

memoria

L'area di esecuzione ad alta velocità contenuta nel computer, dove i dati sono contenuti, copiati e caricati.

menu

Un elenco di opzioni dal quale l'utente può selezionare l'operazione da eseguire.

messaggio d'errore

Un'indicazione che informa in merito ad un difetto di funzionamento dell'hardware o del software o al tentativo di inserimento di dati illeciti.

metodi di accesso

Tecniche e programmi utilizzati per lo spostamento di dati tra memoria principale ed unità periferiche di input/output.

microprocessore

Il processore centrale (CPU) di un computer.

modem

Acronimo di modulatore-demodulatore. La funzione del modem è di convertire i dati presenti in un computer in segnali analogici in modo tale da poterli trasmettere attraverso i cavi telefonici, o di convertire i segnali ricevuti tramite cavi telefonici in forma leggibile dal computer.

MS-DOS

Acronimo di Microsoft Disk Operating System.

nascosto (file)

File non osservabile durante una normale ricerca nella directory.

network

Vedere Rete

nidificato (programma o subroutine)

Un programma o una subroutine contenuta in una routine di dimensioni maggiori per permettere la pronta esecuzione o l'accesso di ciascun livello della routine. La nidificazione di cicli chiusi consiste nell'incorporare una routine di istruzioni in un altro ciclo chiuso.

nome del file

Normalmente, il nome assegnato ad un file dall'utente, che viene usato per l'identificazione del file stesso in tutte le operazioni in cui viene successivamente utilizzato.

nullo

A differenza di zero o di spazi (i quali indicano presenza di informazioni), nullo indica l'assenza di qualsiasi tipo di dati o di informazioni. Ad esempio, nel numero 540, lo zero contiene un informazione necessaria.

numerico

Si riferisce a caratteri numerici e non a caratteri alfabetici.

operando

Una quantità o un dato che è parte di un'operazione. Normalmente, un operando viene designato dalla porzione d'indirizzo di un'istruzione, ma può anche essere un risultato, un parametro o un'indicazione del nome o della posizione della successiva istruzione da eseguire.

operatore

Un simbolo che indica un'operazione. Ad esempio, + rappresenta un'addizione, - una sottrazione, x una moltiplicazione, e / una divisione.

orologio incorporato

Un orologio a tempo reale che permette ai programmi di usare l'ora del giorno e la data. Incorporato in MS-DOS, questo orologio permette l'impostazione della struttura temporale del programma. Inoltre, lo si può utilizzare per tenere un calendario personale e calcolare il tempo trascorso in modo automatico.

ottale (sistema di numerazione)

Una rappresentazione di valori o di quantità attraverso numeri ottali, cioè a base 8 anzichè 10 (numeri decimali). Il sistema di numerazione ottale fa uso delle otto cifre seguenti: 0, 1, 2, 3, 4, 5, 6, e 7. Esso viene usato per facilitare l'espressione delle quantità binarie.

output

I risultati emessi dal computer, oppure i dati elaborati.

parallelo (output)

Il metodo con cui tutti i bit di una parola binaria vengono trasmessi simultaneamente.

parametro

1. Una variabile alla quale viene assegnato un valore per uno specifico programma o operazione. E' una caratteristica definibile di un elemento, un'unità periferica o un sistema.
2. Un'istruzione che, aggiunta ad un comando, imposta una determinata azione in sostituzione di quella predefinita, per il comando che segue.

parità

Un extra-bit di codice usato per controllare la precisione dei dati binari dopo il loro trasferimento in o dalla memoria, mediante la somma di un valore pari o dispari al bit attivo di un dato.

parola

L'insieme di bit che forma la più grande unità gestibile dal computer in una sola operazione.

partizione

Un'area posta sul disco fisso che è riservata ad una determinata funzione; ad esempio, per contenere il sistema operativo.

partizione attiva

Una porzione di memoria del computer che ospita il sistema operativo in uso.

periferica (unità)

Un pezzo di hardware che esegue una funzione specifica. Un esempio di unità periferica è una stampante.

pixel

Un singolo punto sul monitor che può essere indirizzato da un unico bit.

porta

Il canale di entrata e di uscita del computer centrale la cui funzione è collegare una linea di comunicazioni o un'altra unità periferica.

precisione doppia (valori a)

L'utilizzo di due parole del computer per rappresentare ciascun numero. Questa tecnica permette l'uso del doppio delle cifre normalmente disponibili e viene usata quando è necessaria estrema precisione nei calcoli.

precisione singola (valore)

L'aritmetica a precisione singola prevede l'uso di una parola per numero, mentre l'aritmetica a precisione doppia ne usa due, e così via. Per i computer con parole di lunghezza variabile, la precisione è il numero delle cifre usato per indicare un numero. Più la precisione è alta, più sono le posizioni decimali.

predefinito (valore)

Un valore che il computer utilizza automaticamente, a meno che vengano fornite istruzioni diverse.

processore centrale (CPU)

Il cuore del sistema del computer, in cui avviene l'esecuzione effettiva delle operazioni sui dati e dei calcoli. Il processore centrale contiene un'unità di controllo che interpreta ed esegue il programma ed un'unità logica aritmetica che esegue calcoli ed operazioni logiche. Inoltre, il processore centrale dirige le informazioni al punto destinato, controlla l'input e l'output e memorizza temporaneamente i dati.

programma

Una serie di istruzioni impostate in forma accettabile per il computer per l'esecuzione di particolari operazioni. Sono programmi i seguenti tipi di software: sistemi operativi, assemblatori, compilatori, interpreti, sistemi per elaborazione testi, programmi di utilità, ecc..

programma applicativo

Un programma di computer creato per soddisfare specifici bisogni dell'utente.

programmi diagnostici

Programmi speciali utilizzati per l'isolamento di difetti dell'hardware.

prompt

Un carattere o un gruppo di caratteri che appare sullo schermo per sollecitare l'inserimento di istruzioni da parte dell'utente.

protezione del file

I dispositivi o le esecuzioni che prevengono l'annullamento in massa dei dati dall'unità di memorizzazione (un dischetto, ad esempio).

punto mobile (aritmetica a)

Un metodo di calcolo in cui il computer o il programma registra e tiene immediatamente conto della posizione del punto della radice. Il programmatore non ha bisogno di occuparsene.

punto mobile (routine a)

Un insieme di istruzioni che permette operazioni matematiche in un computer privo della possibilità di memorizzare automaticamente il punto della radice.

radiante

L'unità di misura naturale dell'angolo tra due semirette intersecanti sull'angolo da una semiretta ad un'altra semiretta intersecante. E' l'angolo di un arco di una circonferenza uguale in lunghezza al raggio del cerchio. Essendo il perimetro di un cerchio uguale a 2π per il raggio, il numero di radianti in un angolo di 360 gradi o in un cerchio completo è di 2π.

radice

Il numero arbitrario di base di un sistema di numeri. Ad esempio, 10 è il numero base del sistema comune di logaritmi e del sistema numerico decimale.

RAM

Memoria ad accesso casuale; RAM è l'acronimo di random-access memory (memoria ad accesso casuale).

reale (numero)

Un numero ordinario, razionale o irrazionale, privo di parte immaginaria.

remoto

Fa riferimento a unità poste in sedi lontane dal computer centrale.

rete

Una configurazione di hardware con più unità sparse, collegate tramite linee di comunicazioni in grado di garantire la condivisione di dati tra i computer nel funzionamento in rete.

reverse

Visualizzazione di caratteri su un fondo opposto alla visualizzazione normale (nero su bianco, ad esempio).

ROM

acronimo di read only memory (memoria di sola lettura).

RS-232

Un'interfaccia di comunicazioni standard tra un modem ed unità terminali che è coerente con lo EIA Standard RS-232.

seriale (output)

Invio dell'output a unità collegate un bit per volta.

sincrono

Un tipo di operazioni di computer in cui l'esecuzione di ciascuna istruzione è controllata da un segnale d'orologio: impulsi uniformemente distanziati separano l'esecuzione di ciascuna operazione. Tale tipo di operazioni può causare ritardi.

sistema

Un insieme di hardware, software e firmware collegati tra di loro per operare come un'unità.

sistema operativo

Un insieme di istruzioni organizzate che sono alla base di tutte le operazioni del computer.

software

Una stringa di istruzioni che impone al computer l'esecuzione di determinate funzioni.

sola lettura (memoria)

Un tipo di memoria contenente istruzioni o dati permanenti. Il computer può accedere a questa memoria soltanto per la lettura dei dati; la scrittura di dati aggiuntivi non è permessa. Per riferirsi alla memoria di sola lettura, viene spesso usata la parola ROM, acronimo di read only memory.

struttura del file

Una rappresentazione concettuale del modo in cui i dati, i record ed i file vengono collegati tra di loro. Per struttura si intende normalmente il modo in cui i dati devono essere memorizzati ed elaborati.

supporto

Materiale fisico sul quale avviene la memorizzazione dei dati. Alcuni esempi sono: nastri magnetici, schede perforate e dischetti.

tempo reale

1. Il tempo effettivo necessario per risolvere un problema.
2. La risoluzione di un problema durante il tempo effettivo in cui un processo fisico ha luogo, in modo che i risultati possano essere usati per dirigere il processo stesso.

troncare

Concludere un calcolo attenendosi a determinate regole. Ad esempio, eliminare la parte decimale di un numero anziché arrotondarla.

unità disco

Un'unità periferica che contiene e gestisce supporti magnetici sui quali il processore centrale scrive e legge dati.

variabile

Una quantità che può assumere uno qualsiasi dei valori di un certo set come conseguenza di una elaborazione di dati.

Indice analitico

F

G

I

K

L

M

N

O

P

R

S

Guida di riferimento

MICROSOFT®
GW-BASIC® Interpreter

Guida di riferimento

Interprete Microsoft® GW-BASIC®

Guida di riferimento

Microsoft Corporation

Introduzione

Questo manuale è un riferimento alfabetico delle istruzioni, funzioni, comandi e variabili di GW-BASIC.

Il nome e la natura di ciascuna istruzione vengono indicati ad inizio pagina, seguiti da:

Funzione	La funzione dell'istruzione
Sintassi	La notazione completa dell'istruzione
Commenti	Informazioni pertinenti sull'istruzione e la conseguente azione di GW-BASIC quando la incontra
Esempi	Un esempio dell'istruzione all'interno di un un programma
Avvertenze	Qualsiasi speciale informazione riguardante l'istruzione

ABS (funzione)

Funzione

Restituisce il valore assoluto dell'espressione *n*.

Sintassi

ABS(*n*)

Commenti

n deve essere un'espressione numerica.

Esempi

```
PRINT ABS(7*(-5))
35
ok
```

Questa istruzione determina la stampa sullo schermo di *35*.

ASC (funzione)

Funzione

Restituisce un valore numerico che rappresenta il codice ASCII per il primo carattere della stringa X$.

Sintassi

ASC(X$)

Commenti

Se x$ è nullo, viene restituito l'errore `Chiamata di funzione illegale`.

Se x$ inizia con una lettera maiuscola, il valore restituito può variare da 65 a 90.

Se x$ inizia con una lettera minuscola, il valore restituito può variare da 97 a 122.

I numeri che vanno da 0 a 9 restituiscono i valori da 48 a 57, nello stesso ordine.

Si veda la funzione CHR$ per la conversione da ASCII a stringhe.

Per i codici ASCII consultare l'appendice C nel *Manuale dell'utente*.

Esempi

```
10 x$="TEN"
20 PRINT ASC(X$)
RUN
         84
Ok
```

84 è il codice ASCII della lettera T.

ATN (funzione)

Funzione

Restituisce l'arcotangente di x, con x è espresso in radianti.

Sintassi

ATN(x)

Commenti

Il risultato è nell'intervallo di valori che va da $-\pi/2$ a $\pi/2$.

L'espressione *x* può essere di qualsiasi tipo numerico. La valutazione di ATN viene eseguita in precisione singola a meno che non venga usato il parametro /d all'esecuzione di GW-BASIC.

Per la conversione da gradi a radianti, moltiplicare per $\pi/180$.

Esempi

```
10 INPUT X
20 PRINT ATN(X)
RUN
? 3
1.249046
Ok
```

Stampa l'arcotangente di 3 radianti (1.249046).

AUTO (comando)

Funzione

Genera ed incrementa automaticamente i numeri di riga ogni volta che il tasto RITORNO viene premuto.

Sintassi

AUTO [*numeroriga*][,[*incremento*]]
AUTO .[,[*incremento*]]

Commenti

AUTO è utile per l'inserimento di righe di programma in quanto evita la digitazione dei numeri di riga.

AUTO inizia la numerazione al *numeroriga* ed incrementa ogni numero di riga seguente di *incremento.* Il valore predefinito per entrambi i parametri è 10.

Il punto (.) può essere usato in sostituzione del *numeroriga* per indicare la riga corrente.

Se il *numeroriga* viene fatto seguire da una virgola (,) e l'*incremento* non viene specificato, viene utilizzato l'ultimo incremento specificato in un comando AUTO.

Se AUTO genera un numero di riga già in uso, accanto a quel numero appare un asterisco che avverte che qualsiasi inserimento di dati verrà posto in sostituzione della riga esistente. Comunque, premendo RITORNO subito dopo l'asterisco, si salva la riga e si genera il numero di riga seguente.

AUTO viene concluso premendo CTRL-BREAK o CTRL-C. In questo modo GW-BASIC ritorna al livello di comando.

Avvertenza La riga nella quale avviene l'inserimento di CTRL-BREAK o CTRL-C non viene salvata. Per assicurarsi di salvare tutto il testo desiderato, usare CTRL-BREAK e CTRL-C soltanto in righe vuote.

Esempi

```
AUTO 100,50
```

Genera i numeri di riga 100, 150, 200 e così via.

```
AUTO
```

Genera numeri di riga 10, 20, 30, 40, e così via.

BEEP (istruzione)

Funzione

Emette il suono da un altoparlante ad 800 Hz (800 cicli al secondo) per un quarto di secondo.

Sintassi

BEEP

Commenti

BEEP, CTRL-G, e PRINT CHR$(7) producono lo stesso effetto.

Esempi

```
2340 IF X>20 THEN BEEP
```

Se X oltrepassa il limite fissato, il computer emette un suono.

BLOAD (comando)

Funzione

Carica un file di immagine in qualsiasi punto della memoria dell'utente.

Sintassi

BLOAD *nomefile*[,*offset*]

Commenti

nomefile è una espressione alfanumerica valida contenente l'unità periferica ed il nome del file.

offset è un'espressione numerica valida che rientra nell'intervallo di valori che va da 0 a 65535. Si tratta dell'offset nel segmento dichiarato dall'ultima istruzione DEF SEG, dal quale deve avere inizio il caricamento.

Se l'offset viene omesso, viene utilizzato l'offset specificato al comando BSAVE; ciò significa che il file viene caricato nella stessa posizione in cui era stato salvato.

Avvertenza BLOAD non esegue il controllo dell'intervallo dell'indirizzo. Il caricamento è possibile in qualsiasi punto della memoria. Non è permesso invece, il caricamento sullo spazio di stack di GW-BASIC, su un programma GW-BASIC, oppure sulla zona di variabili di GW-BASIC.

Anche se sono utili per caricare e salvare programmi in linguaggio macchina, BLOAD e BSAVE non si limitano ad eseguire questa funzione. L'istruzione DEF SEG permette di specificare qualsiasi segmento come origine o destinazione di BLOAD e BSAVE. Ad esempio, ciò permette la lettura o la scrittura del buffer di schermo sul dischetto. BLOAD e BSAVE sono utili per salvare e visualizzare immagini grafiche.

Esempi

```
10 DEF SEG=&HB800
20 BLOAD"GRAFICO",0
```

(Questo esempio potrebbe non funzionare in alcune modalità di schermo.)

L'istruzione DEF SEG alla riga 10 punta il segmento al buffer di schermo.

L'istruzione DEF SEG alla riga 10 e l'offset 0 in riga 20 garantiscono l'uso dell'indirizzo corretto.

Il comando BLOAD alla riga 20 carica il file chiamato GRAFICO nel buffer di schermo.

Avvertenza L'esempio di BSAVE nella sezione seguente illustra come il file GRAFICO viene salvato.

BSAVE (comando)

Funzione

Salva porzioni di memoria utente nell'unità periferica specificata.

Sintassi

BSAVE *nomefile,offset,lunghezza*

Commenti

nomefile è un'espressione alfanumerica valida contenente il nome del file.

offset è un'espressione numerica valida che rientra nell'intervallo di valori numerici che va da 0 a 65535. Si tratta dell'offset nel segmento dichiarato dall'ultima istruzione DEF SEG, dal quale deve avere inizio il salvataggio.

lunghezza è un'espressione numerica valida che rientra nell'intervallo di valori numerici che va da 0 a 65535; specifica la lunghezza dell'immagine di memoria da salvare.

Se *nomefile* è inferiore ad un carattere, l'operazione viene arrestata ed appare il messaggio d'errore `Numero di file errato.`

Eseguire l'istruzione DEF SEG prima di BSAVE. L'ultimo indirizzo conosciuto di DEG SEG viene sempre usato per l'operazione di memorizzazione.

L'istruzione DEF SEG deve essere usata per l'impostazione dell'indirizzo di segmento all'inizio del buffer di schermo. L'offset 0 insieme ad una lunghezza di 16384 specifica che l'intero buffer di schermo di 16K deve essere salvato.

Esempi

```
10 DEF SEG=&HB800
20 BSAVE"GRAFICO",0,16384
```

L'istruzione DEF SEG alla riga 10 punta il segmento al buffer di schermo.

Il comando BSAVE alla riga 20 salva il buffer di schermo nel file chiamato GRAFICO.

CALL (istruzione)

Funzione

Chiama una subroutine in linguaggio di assemblaggio (o macchina).

Sintassi

CALL*numvar*[(*variabili*)]

Commenti

numvar è il punto di partenza in memoria della subroutine, individuata da un offset nel segmento corrente.

L'opzione *variabili* rappresenta le variabili o le costanti, separate da virgole e racchiuse in parentesi, che devono essere passate alla routine.

L'istruzione CALL è consigliata per i programmi in linguaggio di assemblaggio interfacciale con GW-BASIC. Sebbene anche la funzione USR possa essere usata, CALL è compatibile con più linguaggi, produce istruzioni d'origine più leggibili ed è in grado di passare argomenti multipli.

L'invocazione dell'istruzione CALL comporta quanto segue:

- Ciascuna posizione del parametro nella variabile viene spostata sopra lo stack. La posizione del parametro è un offset di 2-byte all'interno del segmento di dati di GW-BASIC.
- Il segmento delle istruzioni dell'indirizzo di ritorno e l'offset vengono spostati sopra lo stack.
- Il controllo viene trasferito alla routine dell'utente dall'indirizzo di segmento fornito nell'ultima istruzione DEF SEG e dall'offset indicato nel nome della variabile.
- A questo punto la routine dell'utente ha il controllo. Si possono fornire richiami ai parametri spostando il puntatore di stack (SP) al puntatore di base (BP) e sommando un offset positivo a BP.
- La routine chiamata potrebbe distruggere il contenuto di qualsiasi registro.

- Il programma chiamato deve conoscere il numero dei parametri passati. Si può fare riferimento ai parametri sommando un offset positivo al BP, nell'ipotesi in cui la routine chiamata abbia spostato il puntatore di stack corrente in BP (cioè MOV BP,SP).
- Il programma chiamato deve conoscere il tipo di variabile dei parametri numerici passati.
- La routine che viene chiamata deve eseguire un RET *n*, in cui *n* è il numero dei parametri nella variabile moltiplicato per 2. Ciò è necessario per impostare lo stack al punto d'inizio della sequenza di chiamata.
- Dei valori vengono restituiti a GW-BASIC mediante l'inclusione nell'elenco di argomenti del nome della variabile che deve ricevere il risultato.
- Se l'argomento è una stringa, l'offset del parametro punta ai tre byte chiamati **descrittori di stringa**. Il byte 0 del descrittore di stringa contiene la lunghezza della stringa (da 0 a 255). I byte 1 e 2 sono rispettivamente gli otto bit più in basso e più in alto dell'indirizzo di partenza della stringa, nella stringa stessa.
- Se l'argomento è una stringa letterale nel programma, il descrittore di stringa punta al testo del programma. Attenzione a non distruggere od alterare un programma in questo modo. Per evitare risultati imprevedibili, aggiungere +"" alla stringa letterale nel programma, come nel seguente esempio:

  ```
  20 A$="BASIC"+""
  ```

  Questo impone la copia della stringa letterale nello spazio di stringa. Ora la stringa può essere modificata senza influenzare il programma.

Avvertenza L'alterazione delle stringhe attraverso routine dell'utente è permessa, ma bisogna evitare di cambiarne la lunghezza. Se la lunghezza delle stringhe viene modificata da routine esterne, GW-BASIC non può cancellarle correttamente.

Per ulteriori informazioni riguardanti l'istruzione CALL e la funzione URS, si veda l'appendice D nel *Manuale dell'utente*.

Esempio 1

```
100 DEF SEG=&H2000
110 ARK=0
120 CALL ARK(A,B$,C)
.
.
.
```

La riga 100 imposta il segmento a 2000 esadecimale. ARK viene impostato a zero cosicché la chiamata eseguirà la subroutine in posizione 2000:0.

Esempio 2

La sequenza di linguaggio di assemblaggio 8086 che segue dimostra l'accesso ai parametri passati e memorizzati nella variabile C:

```
PUSH PB
MOV PB,PS          ;Inserisce la posizione corrente dello
                    stack in PB.
MOV BX,8[PB]       ;Inserisce l'indirizzo del descrittore
                    di B$.
MOV CL,[BX]        ;Inserisce la lunghezza di B$ i CL.
MOV DX,1 [BX]      ;Inserisce l'indirizzo del testo di B$
                    in DX.
.
.
.
MOV SI,10 [PB]     ;Inserisce l'indirizzo di A in SI.
MOV DI,6 [PB]      ;Porta il puntatore di C in DI.
MOVSW              ;Memorizza la variabile A in C.
RET 6              ;Ripristina lo stack e ritorna.
```

MOVSW copia soltanto due byte. Ciò è sufficiente se le variabili A e C sono intere. Se le variabili sono a precisione singola bisogna copiare quattro byte; se sono a precisione doppia bisogna copiarne otto.

Esempio 3

```
100 DEF SEG=&H2000
110 ACC=&H7FA
120 CALL ACC(A,B$,C)
.
.
.
```

La riga 100 imposta il segmento a 2000 esadecimale. Il valore della variabile ACC viene aggiunto all'indirizzo come parola bassa dopo che il valore DEF SEG viene spostato di quattro bit a sinistra (questa è una funzione del microprocessore, non di GW-BASIC). Alla riga 110, ACC viene impostato a &H7FA, in modo che la chiamata ad ACC esegua la subroutine alla posizione 2000:7FA esadecimale (indirizzo assoluto 207FA esadecimale).

CDBL (funzione)

Funzione

Converte x in un numero a precisione doppia.

Sintassi

CDBL(x)

Commenti

x deve essere un'espressione numerica.

Esempio

```
10 A=454.67
20 PRINT A;CDBL(A)
RUN
454.67 454.6700134277344
Ok
```

Questa istruzione stampa la versione a precisione doppia del valore a precisione singola memorizzato nella variabile A.

In questo esempio gli ultimi undici numeri contenuti nel numero a precisione doppia non hanno alcun significato, dal momento che A era stata precedentemente definita sulla base di due soli valori decimali.

Avvertenza Per la conversione dei numeri in numeri interi ed a precisione singola, rispettivamente, consultare le funzioni CINT e CSNG.

CHAIN (istruzione)

Funzione

Trasferisce il controllo al programma specificato e gli passa (concatena) le variabili dal programma corrente.

Sintassi

CHAIN[MERGE] *nomefile*[,[*riga*][,[**ALL**][,**DELETE** *intervallo*]]]

Commenti

MERGE sovrappone il programma chiamato al programma corrente.

Avvertenza Se deve essere unito, il programma chiamato deve essere un file ASCII (precedentemente salvato con l'opzione a). Si veda il comando MERGE.

nomefile rappresenta il nome del programma chiamato per la concatenazione. A meno che non ne venga specificata un'altra, viene usata l'estensione .BAS.

riga rappresenta il numero di riga o l'espressione che corrisponde al numero di riga nel programma chiamato. *riga* costituisce il punto di partenza per l'esecuzione del programma chiamato. Ad esempio, quanto segue inizia l'esecuzione di PROG1 alla riga 1000:

```
10 CHAIN "PROG1",1000
```

Se *riga* viene omesso, l'esecuzione inizia dalla prima riga.

Il comando RENUM non ha alcun effetto su *riga*. Ha invece effetto sui numeri di riga nell'intervallo specificato.

ALL specifica che tutte le variabili contenute nel programma corrente devono essere concatenate al programma chiamato. Ad esempio:

```
20 CHAIN "PROG1",1000,ALL
```

Se l'opzione ALL viene omessa, il programma corrente deve contenere un'istruzione COMMON che elenchi le variabili che devono essere passate.

CHAIN esegue RESTORE prima di eseguire il programma al quale deve essere concatenato. L'istruzione READ inserisce poi il primo elemento nell'istruzione DATA. La lettura del programma in concatenazione non riprende dal punto in cui era stata interrotta.

Dopo che una sovrapposizione viene eseguita ed usata per una funzione specifica, è di solito preferibile cancellarla in modo da fare posto ad un'altra sovrapposizione. Per eseguire ciò, usare il comando DELETE.

L'istruzione CHAIN insieme al comando MERGE lascia i file aperti e conserva le opzioni di base correnti.

Se il comando MERGE viene omesso, l'impostazione dell'OPTION BASE viene conservata, e CHAIN non salvaguarda alcun tipo di variabile o di funzione definita dall'utente per l'uso nel programma concatenato. Cioè, qualsiasi istruzione DEFINT, DEFSNG, DEFDBL, DEFSTR o DEF FN contenente variabili condivise deve essere rienunciata nel programma concatenato.

Quando si usa il comando MERGE, bisogna porre le funzioni definite dall'utente prima di qualsiasi istruzione CHAIN o MERGE nel programma. Altrimenti, a completamento dell'operazione di unione, esse saranno indefinite.

CHDIR (comando)

Funzione

Passa da una directory di lavoro ad un'altra.

Sintassi

CHDIR *nomepercorso*

Commenti

nomepercorso è una stringa lunga fino a 63 caratteri.

Per rendere FATTURE la directory di lavoro sull'unità disco A: e BILANCIO la directory di lavoro sull'unità disco B:, supponendo che A: sia l'unità disco corrente, digitare i comandi seguenti:

```
CHDIR "FATTURE"
CHDIR "B:BILANCIO"
```

CHR$ (funzione)

Funzione

Converte un codice ASCII nel carattere corrispondente.

Sintassi

CHR$(*n*)

Commenti

n è un valore da 0 a 255.

CHR$ viene comunemente usato per inviare un carattere speciale al terminale o alla stampante. Ad esempio, si può inviare CHR$(7) per determinare l'emissione di un suono prima di un messaggio d'errore; oppure si può inviare un cambio pagina, CHR(12), alla stampante.

Per la conversione da caratteri ASCII a codici numerici, consultare la funzione ASC.

I codici ASCII sono elencati nell'appendice C del *Manuale dell'utente* di GW-BASIC.

Esempi

```
PRINT CHR$(66);
B
Ok
```

Questa istruzione stampa il codice ASCII del carattere 66, cioè la lettera maiuscola B.

```
PRINT CHR$(13);
```

Questa istruzione stampa un ritorno di riga.

CINT (funzione)

Arrotonda i numeri con porzioni frazionali al numero intero successivo.

Sintassi

CINT(*x*)

Commenti

Se *x* non rientra nell'intervallo numerico che va da -32768 a 32767, appare il messaggio d'errore `Valore troppo elevato.`

Consultare le funzioni FIX ed INT, che restituiscono numeri interi.

Esempi

```
PRINT CINT(45.67)
46
Ok
```

45.67 viene arrotondato a 46.

Avvertenza Per convertire i numeri in dati a precisione doppia ed a precisione singola, consultare rispettivamente le funzioni CDBL e CSNG.

CIRCLE (istruzione)

Funzione

Durante l'uso della modalità grafica, questa istruzione disegna un cerchio, un'ellisse e degli angoli sullo schermo.

Sintassi

CIRCLE(*xcentro,ycentro*),*raggio*[,[*colore*][,[*inizio*],[*fine*] [,*proporzione*]]]

Commenti

xcentro e ycentro sono le coordinate del centro dell'ellisse, mentre *raggio* è il raggio (misurato lungo l'asse principale) dell'ellisse. Le quantità *xcentro* e *ycentro* possono essere espressioni. Gli attributi di centro possono usare coordinate assolute o relative.

colore specifica il colore dell'ellisse. Il valore dipende dalla modalità corrente dello schermo.

Per ulteriori informazioni riguardanti l'uso dei colori in modalità di schermo diverse, si vedano le istruzioni COLOR e SCREEN.

Nella modalità ad alta risoluzione, 0 indica il nero mentre 1 indica il bianco. Il valore predefinito per la modalità ad alta risoluzione è 1.

I parametri di angolo, *inizio* e *fine*, sono argomenti del radiante tra -2*π e 2*π che specificano l'inizio e la fine dell'ellisse. Se *inizio* o *fine* sono negativi, l'ellisse viene collegata nel punto centrale tramite una riga, e gli angoli vengono trattati come se fossero positivi (notare che ciò è diverso da sommare 2*π).

proporzione descrive la proporzione tra il raggio x ed il raggio y (x:y). Il valore predefinito della proporzione dipende dalla modalità dello schermo. Un cerchio visivo viene comunque fornito indipendentemente dalla modalità grafica in cui ci si trova, utilizzando un valore standard di 4:3.

Se la proporzione è inferiore ad 1, il raggio viene fornito in x-pixel. Se invece la proporzione è superiore ad 1, il raggio viene fornito in y-pixel.

In molti casi, la proporzione 1 dà ellissi migliori nella modalità a risoluzione media. Ciò permette anche una maggiore rapidità nel disegno della figura. L'angolo iniziale può essere inferiore all'angolo finale.

Esempio 1

```
10 SCREEN1: CIRCLE(100,100), 50
```

Questo traccia un cerchio dal raggio di 50, con centro nel punto di coordinate grafiche x=100 e y=100.

Esempio 2

```
1  ' Questo disegnerà 17 ellissi
10 CLS
20 SCREEN 1
30 FOR R=160 TO 0 STEP-10
40 CIRCLE (160,100),R,,,,5/18
50 NEXT
```

Esempio 3

```
10 'Questo disegnerà 5 sfere
20 GOTO 160
50 IF VERT GOTO 100
60 CIRCLE (X,Y),R,C,,,.07
70 FOR I = 1 TO 5
80 CIRCLE (X,Y),R,C,,,I*.2:NEXT I
90 IF VERT THEN RETURN
100 CIRCLE (X,Y),R,C,,,1.3
110 CIRCLE (X,Y),R,C,,,1.9
120 CIRCLE (X,Y),R,C,,,3.6
130 CIRCLE (X,Y),R,C,,,9.8
140 IF VERT GOTO 60
150 RETURN
160 CLS:SCREEN 1:COLOR 0,1:KEY OFF:VERT=0
170 X=160:Y=100:C=1:R=50:GOSUB 50
180 X=30:Y=30:C=2:R=30:GOSUB 50
190 X=30:Y=169:GOSUB 50
200 X=289:Y=30:GOSUB 50
210 X=289:Y=30:GOSUB 50
220 LINE (30,30) - (289,169),1
230 LINE (30,169) - (289,30),1
240 LINE (30,169) - (289,30),1,B
250 Z$=INKEY$: IF Z$="" THEN 250
RUN
```

CLEAR (comando)

Funzione

Imposta tutte le variabili numeriche a zero, annulla tutte le variabili alfanumeriche e chiude tutti i file aperti. Le opzioni determinano la fine della memoria e riservano per GW-BASIC la quantità di spazio per stringhe e di stack disponibile.

Sintassi

CLEAR[,[*espressione1*]][,*espressione2*]]

Commenti

Se specificata, *espressione1* rappresenta la posizione in memoria che determina il numero massimo di byte disponibili per GW-BASIC.

espressione2 riserva spazio di stack per GW-BASIC. Come valore predefinito viene utilizzata la dimensione dello spazio di stack precedente. Quando GW-BASIC viene inizialmente eseguito, lo spazio di stack viene impostato al valore minore tra 512 byte e un ottavo della memoria disponibile.

GW-BASIC assegna spazio di stringa dinamicamente. Un errore di `Memoria per stringhe esaurita` si verifica soltanto nel caso che non vi sia abbastanza memoria residua disponibile per GW-BASIC.

Il comando CLEAR:

- Chiude tutti i file
- Libera tutte le variabili comuni (COMMON) e quelle definite dall'utente
- Reimposta lo spazio di stack e di stringa
- Libera tutti i buffer di disco
- Disattiva qualsiasi suono
- Reimposta il suono di primo piano
- Disattiva PEN
- Disattiva STRING
- Disattiva ON ERROR

Esempi

```
CLEAR
```

Azzera le variabili ed annulla tutte le stringhe.

```
CLEAR 32768
```

Azzera le variabili, annulla le stringhe, protegge la memoria al di sopra di 32768, e non cambia lo spazio di stack.

```
CLEAR ,,2000
```

Azzera le variabili, annulla le stringhe, assegna 2000 byte per spazio di stack ed utilizza tutta la memoria disponibile nel segmento.

```
CLEAR ,32768,2000
```

Azzera le variabili, annulla le stringhe, protegge la memoria al di sopra di 32768 ed assegna 2000 byte per spazio di stack.

CLOSE (istruzione)

Funzione

Conclude l'input/output ad un file di disco o ad un dispositivo periferico.

Sintassi

CLOSE [[#]*numerofile*[,[#]*numerofile*]...]

Commenti

numerofile rappresenta il numero sotto il quale il file è stato aperto.

L'associazione tra un particolare file od un dispositivo ed il numero del file viene automaticamente eliminata all'esecuzione dell'istruzione CLOSE. La riapertura del file o del dispositivo può poi essere effettuata anche utilizzando un numero diverso.

L'istruzione CLOSE priva della specificazione del numero di file chiude tutti i file ed i dispositivi aperti.

L'istruzione CLOSE inviata ad un file o ad un dispositivo aperto per output sequenziale, scrive il buffer finale di output sul file o dispositivo.

Le istruzioni END, NEW, RESET, SYSTEM, o RUN e LOAD (senza l'opzione **r**) chiudono automaticamente tutti i file o dispositivi. STOP non determina la chiusura dei file.

Esempio

```
250 CLOSE
```

Questa istruzione chiude tutti i dispositivi ed i file aperti.

```
300 CLOSE 1,#2,#3
```

Chiude tutti i file ed i dispositivi associati con i numeri di file 1, 2 e 3.

CLS (istruzione)

Funzione

Libera lo schermo.

Sintassi

CLS [*n*]

Commenti

n corrisponde ad uno dei valori seguenti:

Valore di n	*Effetto*
0	Libera lo schermo da tutti i testi e da tutti i grafici
1	Libera soltanto la porta di visualizzazione dei grafici
2	Libera soltanto la finestra di testo

Se la porta di visualizzazione dei grafici è attiva, CLS senza argomento libera soltanto la porta di visualizzazione. Se la porta di visualizzazione dei grafici non è attiva, CLS libera la finestra di testo.

Se lo schermo è in modalità alfa, la pagina attiva viene liberata e lo schermo appare nel colore di fondo selezionato (consultare le istruzioni SCREEN e COLOR).

Se lo schermo è in modalità grafica, l'intero buffer di schermo viene liberato.

La liberazione dello schermo può anche essere effettuata premendo CTRL-HOME, oppure cambiando la modalità di schermo con le istruzioni SCREEN e WIDTH.

CLS riporta il cursore sull'angolo superiore sinistro dello schermo ed imposta l'ultimo punto a cui si è fatto riferimento al centro dello schermo.

Se si è usata l'istruzione VIEW, CLS libera soltanto l'ultima porta di visualizzazione specificata.

Esempi

```
1 CLS
```

Libera lo schermo.

COLOR (istruzione)

Funzione

Seleziona colori di visualizzazione

Sintassi

COLOR [*primopiano*][,[*fondo*][,*bordo*]]
COLOR [*fondo*][,[*setcolori*]]
COLOR [*primopiano*][,[*fondo*]]

Commenti

Generalmente, COLOR permette di selezionare i colori di fondo e di primo piano per la visualizzazione. In SCREEN 0 si può anche selezionare un colore per il bordo. In SCREEN 1 la selezione di colori di primo piano non è permessa. E' invece permessa la selezione di uno dei due set di quattro colori per l'uso con le istruzioni grafiche. Le sintassi e gli effetti che vengono impiegati nelle varie modalità di schermo sono descritti sotto:

Modalità	*Effetto*
SCREEN 0	Modifica i colori predefiniti di fondo e di primo piano del testo corrente e i bordi dello schermo. Il colore di *primopiano* deve essere un'espressione intera tra 0 e 31. Esso viene usato per determinare il colore di primo piano nella modalità di testo, cioè il colore predefinito del testo. E' possibile selezionare fino a sedici colori, usando i numeri interi da 0 a 15 . E' inoltre possibile selezionare una versione intermittente per ciascun colore aggiungendo 16 al numero del colore; ad esempio, il colore intermittente di 7 è uguale a 7 + 16, o 23. Per cui, i valori leciti per l'opzione *primopiano* sono gli interi da 0 a 31.

Il colore di *fondo* deve essere un'espressione intera tra 0 e 7, ed è il colore di fondo di ogni carattere di testo. I colori intermittenti non sono permessi.

Il colore di *bordo* è un'espressione intera tra 0 e 15, ed è il colore usato durante il disegno dei bordi dello schermo. I colori intermittenti non sono permessi.

Se all'istruzione COLOR non viene fornito nessun argomento, il colore predefinito per *fondo* e *bordo* è il nero (il colore 0), mentre quello per *primopiano* è quello descritto nelle pagine di riferimento dell'istruzione SCREEN.

SCREEN 1

In modalità 1, l'istruzione COLOR possiede una sintassi unica comprendente un argomento di *setcolori*, che è un'espressione intera dispari o pari. Questo argomento determina il set di colori da usare quando si visualizzano particolari numeri di colore.

Per le configurazioni di hardware che non dispongono di un adattatore grafico avanzato IBM (EGA), le impostazioni di colore predefinite per il parametro *setcolori* sono equivalenti alle seguenti:

COLOR ,0	Equivale alle tre istruzioni PALETTE che seguono 1 = verde, 2 = rosso, 3 = giallo
COLOR ,1	Equivale alle tre istruzioni PALETTE che seguono 1 = turchese, 2 = magenta, 3 = bianco intenso

Usando l'adattatore grafico avanzato (EGA), le impostazioni di colori predefinite per il parametro *setcolori* sono equivalenti alle seguenti:

COLOR ,0	Equivale alle tre istruzioni PALETTE che seguono
PALETTE 1,2	attributo 1 = colore 3 (verde)
PALETTE 2,4	attributo 2 = colore 5 (rosso)
PALETTE 3,6	attributo 3 = colore 6 (marrone)
COLOR ,1	Equivale alle tre istruzioni PALETTE che seguono
PALETTE 1,3	attributo 1 = colore 3 (turchese)
PALETTE 2,5	attributo 2 = colore 5 (magenta)
PALETTE 3,7	attributo 3 = colore 15 (bianco)

Si avverte che l'istruzione COLOR ignora le istruzioni PALETTE precedenti.

SCREEN 2

Nessun effetto. Se COLOR viene usato in questa modalità, appare il messaggio d'errore `Chiamata di funzione illegale.`

SCREEN 7-SCREEN 10

In queste modalità, la specificazione di un colore per il *bordo* non è permessa. Il fondo dei grafici viene dato dal numero di colore di *fondo*, che deve essere composto da un intervallo di numeri di colore validi adeguati alla modalità dello schermo. Per una spiegazione più dettagliata, consultare le pagine di riferimento dell'istruzione SCREEN. L'argomento del colore di *primopiano* è il colore predefinito per tracciare righe.

Gli argomenti che non fanno parte dell'intervallo numerico valido determinano errori di `Chiamata di funzione illegale.`

Il colore di primo piano può essere identico a quello di fondo; ciò rende invisibili i caratteri visualizzati. Per tutte le configurazioni di visualizzazione hardware e tutte le modalità di schermo, il colore di fondo predefinito è il nero, cioè il numero di colore 0.

Con un adattatore grafico avanzato, l'istruzione PALETTE permette flessibilità nell'assegnazione di colori di visualizzazione diversi per *primopiano*, *fondo* e *bordo*. Per ulteriori dettagli, si vedano le pagine di riferimento dell'istruzione PALETTE.

Per ulteriori informazioni, si vedano CIRCLE, DRAW, LINE, PAINT, PALETTE, PRESET, PSET, SCREEN

Esempi

La serie di esempi che segue mostra le istruzioni COLOR e gli effetti nelle diverse modalità di schermo:

```
SCREEN 0
COLOR  1, 2, 3   'primo piano=1, fondo=2, bordo=3
SCREEN 1
COLOR  1,0       'primo piano=1, numero set di colori
                  pari
COLOR  2,1       'primo piano=2, numero set di colori
                  dispari
SCREEN 7
COLOR  3,5       'primo piano=3, fondo=5
SCREEN 8
COLOR  6,7       'primo piano=6, fondo=7
SCREEN 9
COLOR  1,2       'primo piano=1, fondo=2
```

COM(n) (istruzione)

Funzione

Abilita o disabilita il trapping delle attività di comunicazione all'adattatore di comunicazioni specificato.

Sintassi

COM(*n*) ON
COM(*n*) OFF
COM(*n*) STOP

Commenti

n è il numero dell'adattatore di comunicazioni, 1 o 2.

Eseguire l'istruzione COM(*n*) ON prima dell'istruzione ON COM(*n*) per permettere il trapping. Se, dopo COM(*n*) ON, viene specificato un numero diverso da zero nell'istruzione ON COM(*n*), BASIC controlla tutte le nuove istruzioni per verificare l'eventuale inserimento di caratteri nell'adattatore di comunicazioni.

Con COM(*n*) OFF non avviene alcun trapping, e di conseguenza tutte le attività di comunicazione vengono perse.

Con COM(*n*) STOP, non avviene alcun trapping. Tuttavia, se vi sono delle comunicazioni, esse vengono memorizzate in modo che all'esecuzione di COM(*n*) ON si verifichi un trapping immediato.

COMMON (istruzione)

Passa variabili ad un programma concatenato.

Sintassi

COMMON *variabili*

Commenti

variabili rappresenta una o più variabili, separate da virgole, da passare al programma concatenato.

L'istruzione COMMON viene usata insieme all'istruzione CHAIN.

Le istruzioni COMMON possono apparire in qualsiasi punto del programma; è comunque preferibile che esse appaiono all'inizio.

In un programma può apparire qualsiasi numero di istruzioni COMMON. Non può invece apparire la stessa variabile in più di un'istruzione COMMON. Per passare le variabili con l'istruzione CHAIN, usare l'opzione ALL ed omettere l'istruzione COMMON.

Porre parentesi dopo il nome della variabile per indicare le variabili di matrice.

Esempi

```
100 COMMON A, B, C, D(),G$
110 CHAIN "A:PROG3"
```

Questo esempio mostra il concantenamento del programma PROG3 sull'unità disco A: ed il passaggio della matrice D con le variabili A, B, C e la stringa G$.

CONT (comando)

Funzione

Continua l'esecuzione del programma dopo un'interruzione.

Sintassi

CONT

Commenti

Riprende l'esecuzione del programma interrotto da CTRL-BREAK, STOP o END. L'esecuzione viene ripresa al punto di interruzione. Se l'interruzione è avvenuta durante l'istruzione INPUT, l'esecuzione continua dopo la riproduzione del prompt.

CONT è utile per il debugging, in quanto permette l'impostazione di punti di interruzione con l'istruzione STOP, la modifica delle variabili con istruzioni dirette e la continuazione dell'esecuzione del programma al punto di interruzione o da un punto specifico (individuato attraverso GOTO).

Se una riga di programma viene modificata, CONT viene invalidato.

COS (funzione)

Funzione

Restituisce il coseno dell'intervallo di *x*.

Sintassi

COS(*x*)

Commenti

x deve essere il radiante. COS è la funzione trigonometrica del coseno. Per la conversione da gradi a radianti, moltiplicare per $\pi/180$.

COS(*x*) viene calcolato in precisione singola a meno che sia stato usato il parametro **/d** all'atto dell'esecuzione di GW-BASIC.

Esempio 1

```
10 X=2*COS (.4)
20 PRINT X
RUN
1.842122
Ok
```

Esempio 2

```
10 PI=3.141593
20 PRINT COS(PI)
30 GRADI=180
40 RADIANTI=GRADI*PI/180
50 PRINT COS(RADIANTI)
RUN
-1
-1
OK
```

CSNG (funzione)

Funzione

Esegue la conversione di *x* in un numero a precisione singola.

Sintassi

CSNG(*x*)

Commenti

x deve essere un'espressione numerica (si vedano le funzioni CINT e CDBL).

Esempi

```
10 A>=975.3421222>
20 PRINT A>; CSNG(A>)
RUN
  975.3421222    975.3421
Ok
```

CSRLIN (variabile)

Funzione

Restituisce la posizione di riga corrente del cursore.

Sintassi

y=CSRLIN

Commenti

y rappresenta la variabile numerica che riceve il valore restituito. Il valore restituito varia da 1 a 25.

La variabile CSRLIN restituisce la coordinata verticale del cursore sulla pagina attiva (si veda l'istruzione SCREEN).

x=POS(0) restituisce la posizione della colonna nella quale si trova il cursore. Il valore restituito varia da 1 a 40, o da 1 a 80, a secondo della larghezza dello schermo corrente (si veda la funzione POS).

Esempi

```
10 y=CSRLIN
20 X=POS(0)
30 LOCATE 24,1
40 PRINT "CIAO"
50 LOCATE Y,X
RUN
CIAO
Ok
```

La variabile CSRLIN alla riga 10 registra la riga corrente.

La funzione POS alla riga 20 registra la colonna corrente.

Alla riga 40, l'istruzione PRINT visualizza la parola "CIAO", nella 24ma riga dello schermo.

L'istruzione LOCATE alla riga 50 riporta il cursore sulla riga e sulla colonna d'origine.

CVI, CVS, CVD (funzioni)

Funzione

Convertono i valori alfanumerici in valori numerici.

Sintassi

CVI(*stringa a 2 byte*)
CVS(*stringa a 4 byte*)
CVD(*stringa a 8 byte*)

Commenti

Se si desidera eseguire qualsiasi operazione aritmetica, i valori numerici letti dal disco ad accesso casuale devono essere riconvertiti da stringhe a numeri.

CVI converte una stringa a 2 byte in un numero intero. Il suo complemento è MKI$.

CVS converte una stringa a 4 byte in un numero a precisione singola. Il suo complemento è MKS$.

CVD converte una stringa a 8 byte in un numero a precisione doppia. Il suo complemento è MKD$.

(Si vedano MKI$, MKS$, E MKD$.)

Esempi

```
.
.
.
70 FIELD #1,4 AS N$, 12 AS B$...
80 GET #1
90 Y=CVS(N$)
.
.
.
```

La riga 80 legge un campo dal file numero 1 (il campo di lettura viene definito alla riga 70), e converte i primi quattro byte (N$) in un numero a precisione singola assegnato alla variabile Y.

Dal momento che il numero a precisione singola può contenere fino a sette caratteri ASCII (sette byte), quando un file viene scritto con la conversione MKS$ o letto con la conversione CVS, vengono registrati e salvati fino a tre byte per ciascun numero sul supporto di memorizzazione. Se sono richiesti numeri a precisione doppia, se ne potrebbero salvare anche di più. In questo secondo caso si utilizzerebbero conversioni MKD$ e CVD.

DATA (istruzione)

Funzione

Memorizza le costanti numeriche e alfanumeriche caricate dall'istruzione (o dalle istruzioni) READ del programma.

Sintassi

DATA *costanti*

Commenti

costanti rappresenta le costanti numeriche in qualsiasi formato (a punto fisso, punto mobile, o intere), separate da virgole. Nell'elenco non è permessa alcuna espressione.

Le costanti alfanumeriche nelle istruzioni DATA devono essere racchiuse tra virgolette soltanto se contengono virgole, due punti o spazi significativi.

Le istruzioni DATA non sono eseguibili e possono essere inserite ovunque nel programma. Un'istruzione DATA può contenere tutte le costanti (separate da virgole) che possono trovare posto nella riga. In un programma è possibile utilizzare un numero indeterminato di istruzioni DATA.

Le istruzioni READ hanno accesso alle istruzioni DATA in ordine (in base al numero di riga). I dati contenuti nelle istruzioni DATA possono essere considerati come un elenco continuo di elementi, indipendentemente dal numero di elementi per riga e dalla posizione esatta delle righe nel programma. Il tipo di variabile (numerica o alfanumerica) indicato durante l'uso dell'istruzione READ deve essere coerente con la natura della costante corrispondente nell'istruzione DATA; altrimenti si verifica l'errore `Tipi di carattere contrastanti.`

Le istruzioni DATA possono essere rilette dall'inizio usando l'istruzione RESTORE.

Per ulteriori informazioni ed esempi, si vedano le istruzioni RESTORE e READ.

Esempio 1

```
.
.
.
80 FOR I=1 TO 10
90 READ A(I)
100 NEXT I
110 DATA 3.08,5.19,3.12,3.98,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
.
.
.
```

Questo segmento di programma legge i valori dalle istruzioni DATA nella matrice A. Conseguentemente all'esecuzione, il valore A(1) corrisponde a 3.08, e così via. Le istruzioni DATA (le righe 110 e 120) possono essere poste in qualsiasi parte del programma; anche dopo l'istruzione READ.

Esempio 2

```
5 PRINT
10 PRINT "COMUNE", "PROVINCIA", "CAP"
20 READ C$,P$,A
30 DATA "SERIATE","(BERGAMO)",24068
40 PRINT C$,P$,A
RUN
CITTA'          REGIONE         CAP
SERIATE         BERGAMO         24068
Ok
```

Questo programma legge i dati numerici e alfanumerici dall'istruzione DATA alla riga 30.

DATE$ (istruzione e variabile)

Funzione

Imposta o carica la data corrente.

Sintassi

Come istruzione:

DATE$=v$

Come variabile:

v$=DATE$

Commenti

v$ rappresenta una stringa valida variabile o letterale.

Nell'assegnazione della data le impostazioni valide per v$ sono le seguenti (formato americano):

mm-gg-aa
mm/gg/aa
mm-gg-aaaa
mm/gg/aaaa

Se v$ non è una stringa valida, si produce l'errore `Tipi di carattere contrastanti`.

Se uno dei valori oltrepassa i limiti o è mancante, si produce l'errore `Chiamata di funzione illegale`. Qualsiasi dato precedente viene conservato.

Se DATE$ è l'espressione in un'istruzione LET o PRINT, la data corrente (impostata all'inizializzazione del sistema operativo) viene caricata ed assegnata alla stringa variabile.

Se viene assegnata DATE$ come destinazione ad una stringa, la data corrente viene memorizzata.

Con v$=DATE$, DATE$ restituisce una stringa a 10 caratteri nel formato *mm-gg-aaaa*. *mm* rappresenta i mesi (da 01 a 12), *gg* rappresenta i giorni (da 01 a 31), mentre *aaaa* rappresenta gli anni (da 1980 a 2099).

Esempi

```
v$=DATE$
Ok
PRINT V$
01-01-1986
Ok
```

DEF FN (istruzione)

Funzione

Definisce e nomina una funzione scritta dall'utente.

Sintassi

DEF FN*nome*[(*argomenti*)]=*espressione*

Commenti

nome deve essere un nome di variabile legale. Preceduto da FN, questo nome diventa il nome della funzione.

Il parametro *argomenti* è composto dai nomi delle variabili della definizione della funzione destinati ad essere sostituiti quando la funzione viene chiamata. Gli elementi contenuti nell'elenco sono separati da virgole.

espressione è un'espressione che esegue l'operazione della funzione. E' limitata ad una sola istruzione.

Nell'istruzione DEF FN, gli argomenti servono soltanto a definire la funzione; essi non influenzano le variabili del programma che hanno lo stesso nome. Il nome di una variabile utilizzato nella definizione di una funzione, potrebbe apparire o non apparire nell'argomento. Se appare, il valore del parametro viene fornito quando viene chiamata la funzione. Altrimenti, viene utilizzato il valore corrente della variabile.

Le variabili nell'argomento corrispondono ai valori o alle variabili che devono essere forniti alla chiamata della funzione.

Le funzioni definite dall'utente possono essere di due tipi: numeriche o alfanumeriche. Se nel nome della funzione viene specificato il tipo, il valore dell'espressione viene impostato su tale tipo prima di essere restituito all'istruzione di chiamata. Se viene specificato il tipo nel nome della funzione e il tipo di argomento non corrisponde, si produce l'errore `Tipi di carattere contrastanti.`

Una funzione può essere definita più volte dall'utente ripetendo l'istruzione DEF FN.

L'istruzione DEF FN deve essere eseguita prima che la funzione da essa definita venga chiamata. Se una funzione viene chiamata prima di essere definita, si produce l'errore `Funzione utente non definita.`

L'uso dell'istruzione DEF FN in modalità diretta non è lecito.

Le funzioni ricorsive non sono supportate nell'istruzione DEF FN.

Esempi

```
.
.
.
400 R=1:S=2
410 DEF FNAB(X,Y)=X^3/Y^2
420 T=FNAB(R,S)
.
.
.
```

La riga 410 definisce la funzione utente FNAB. Questa funzione viene poi chiamata alla riga 420. Quando viene eseguita, la variabile T conterrà il valore R al cubo diviso per S al quadrato, cioè 0.25.

DEFINT/SNG/DBL/STR (istruzioni)

Funzione

Impostano i tipi di variabili come numeri interi, a precisione singola, a precisione doppia o come stringhe.

Sintassi

DEF*tipo lettere*

Commenti

tipo rappresenta INT (numero intero), SNG (numero a precisione singola), DBL (numero a precisione doppia) o STR (stringa da 0 a 255 caratteri).

lettere rappresenta delle lettere singole o degli intervalli di lettere dell'alfabeto separati da virgole.

L'istruzione DEF*tipo* indica che i nomi delle variabili che iniziano con le *lettere* specificano un determinato tipo di variabile. Comunque, nella digitazione di una variabile, un simbolo di tipo di carattere (%, !, >, $) ha sempre la precedenza sull'istruzione DEF*tipo*.

Se nessuna istruzioni per l'impostazione del tipo di carattere viene specificata, BASIC suppone che tutte le variabili siano a precisione singola.

Esempi

```
10 DEFDBL L-P
```

In seguito all'esecuzione dell'istruzione contenuta in questa riga, tutte le variabili che iniziano per L, M, N, O, P saranno variabili a precisione doppia.

```
10 DEFSTR A
20 A="120#"
```

In seguito all'esecuzione delle istruzioni contenute in queste ultime due righe, tutte le variabili che iniziano per A saranno variabili alfanumeriche. In questo esempio il simbolo $ non è necessario.

```
10 DEFINT I-N,W-Z
20 W$="120#"
```

In seguito all'esecuzione delle istruzioni contenute in queste due righe, tutte le variabili che iniziano per I, J, K, L, M, N, W, X, Y e Z saranno variabili intere. W$ alla riga 20 imposta una variabile alfanumerica che inizia con la lettera W. Comunque, nella parte restante del programma, la variabile W rimane intera.

DEF SEG (istruzione)

Funzione

Stabilisce l'indirizzo del segmento corrente che verrà usato come riferimento per un'eventuale istruzione BLOAD, BSAVE, CALL, PEEK, POKE o USR.

Sintassi

DEF SEG [*indirizzo*]

Commenti

indirizzo è un'espressione numerica compresa tra 0 e 65535.

L'indirizzo specificato viene salvato ed utilizzato come segmento richiesto dalle istruzioni BLOAD, BSAVE, PEEK, POKE e CALL.

L'inserimento di qualsiasi valore che non rientra nei limiti dell'indirizzo (da 0 a 65535) determina l'errore `Chiamata di funzione illegale`. Il valore precedente non viene in questo caso cambiato.

Se l'opzione *indirizzo* viene omessa, viene utilizzato il segmento di dati di GW-BASIC (DS), che rappresenta perciò il valore iniziale predefinito.

Se l'opzione *indirizzo* viene specificata, deve essere basata sul confine di 16 byte.

Gli indirizzi del segmento vengono spostati di 4 bit verso sinistra. Per ottenere quindi l'indirizzo del segmento, occorre dividere la posizione in memoria per 16.

Per le istruzioni BLOAD, BSAVE, PEEK, POKE o CALL, il valore viene spostato di 4 bit verso sinistra (ciò viene eseguito dal microprocessore anziché da GW-BASIC) per formare l'indirizzo di segmento per un'eventuale istruzione di chiamata (si vedano le istruzioni BLOAD, BSAVE, CALL, PEEK e POKE).

GW-BASIC non esegue ulteriori controlli per assicurare che l'indirizzo di segmento risultante sia valido.

Esempi

```
10 DEF SEG=&HB800
```

Imposta il segmento al buffer di schermo.

```
20 DEF SEG
```

Ripristina il segmento BASIC DS.

Avvertenza DEF e SEG devono essere separati da uno spazio. Altrimenti, GW-BASIC li interpreta come l'istruzione DEFSEG=100, che significa "assegnare il valore 100 alla variabile DEFSEG".

DEF USR (istruzione)

Funzione

Specifica l'indirizzo iniziale di una subroutine in linguaggio d'assemblaggio destinata ad essere chiamata dalla funzione USR.

Sintassi

DEF USR[*n*]=*intero*

Commenti

n può essere qualsiasi cifra compresa tra 0 e 9. Questa cifra corrisponde all'indirizzo della routine utente che viene specificata. Se *n* viene omesso, si assume DEF USR0.

intero rappresenta l'offset di indirizzo della routine utente. Se il programma richiede l'utilizzo di più di 10 routine, DEF USR[*n*] potrebbe figurare nel programma tante volte quante sono necessarie per definire l'indirizzo di partenza di USR[*n*].

Per ottenere l'indirizzo iniziale della routine definita dall'utente, aggiungere il valore del segmento corrente al numero intero.

Quando una subroutine in linguaggio d'assemblaggio viene chiamata, l'esecuzione di GW-BASIC viene sospesa ed il controllo viene trasferito al programma in linguaggio d'assemblaggio. Al termine dell'esecuzione di tale programma, il controllo viene restituito a GW-BASIC al punto d'interruzione.

Esempi

```
.
.
.
190 DEF SEG=0
200 DEF USR0=24000
210 X=USR0(Y^2/2.82)
.
.
.
```

Le righe 190 e 200 impostano l'indirizzo assoluto.

La riga 210 chiama la routine USR posta a quell'indirizzo e passa il valore intero dell'espressione racchiusa tra parentesi al programma dell'utente (si veda USR).

Avvertenza L'istruzione USR viene fornita principalmente per ragioni di compatibilità con altre implementazioni di GW-BASIC. Se questa compatibilità non è rilevante, è consigliabile utilizzare l'istruzione CALL che risulta più versatile.

DELETE (comando)

Funzione

Cancella righe o gruppi di righe di un programma.

Sintassi

DELETE [*numeroriga1*][-*numeroriga2*]
DELETE *numeroriga1*-

Commenti

numeroriga1 rappresenta la prima riga da cancellare.

numeroriga2 invece, rappresenta l'ultima riga da cancellare.

Dopo l'esecuzione del comando DELETE, GW-BASIC ritorna sempre al livello di comando. Bisogna fornire almeno un numero di riga. Altrimenti si determina l'errore `Chiamata di funzione illegale`.

In sostituzione di *numeroriga1* o *2*, si può utilizzare un punto (.) ad indicare la riga corrente.

Esempi

```
DELETE 40
```

Cancella la riga 40.

```
DELETE 40-100
```

Cancella le righe da 40 a 100.

```
DELETE -40
```

Cancella tutte le righe che precedono la riga 40 (compresa).

```
DELETE 40-
```

Cancella tutte le righe da 40 alla fine del programma.

DIM (istruzione)

Funzione

Specifica i valori massimi degli indici di variabili di matrice e assegna la memoria concordemente.

Sintassi

DIM *variabile(indici)*[,*variabile(indici)*]...

Commenti

Se il nome di una variabile viene usato senza l'istruzione DIM, il valore massimo predefinito del suo indice è 10. Se l'indice supera il valore massimo specificato, si produce l'errore `Dimensioni dell'indice errate.`

Il valore massimo assegnabile ad un indice in una matrice è 255.

A meno che non venga diversamente specificato con l'istruzione OPTION BASE, il valore minimo è sempre 0.

Una volta assegnata una dimensione ad una matrice, non le si può riassegnare una dimensione diversa senza prima eseguire le istruzioni CLEAR o ERASE.

L'istruzione DIM imposta tutti gli elementi delle matrici specificate ad un valore iniziale di zero.

Esempi

```
10 DIM A(20)
20 FOR I=0 TO 20
30 READ A(I)
40 NEXT I
```

In quest'esempio vengono lette 21 istruzioni DATA (non mostrate nell'esempio) nel programma e vengono assegnati i valori da A(0) a A(20), in ordine consecutivo. Se la matrice A è a precisione singola (la precisione predefinita) la riga 10 gli assegna 84 byte (4 byte per 21 elementi).

DRAW (istruzione)

Funzione

Disegna una figura.

Sintassi

DRAW *stringa espressione*

Commenti

L'istruzione DRAW combina la maggior parte delle capacità delle altre istruzioni grafiche in un linguaggio per la definizione di oggetti chiamato Linguaggio di Macro Grafiche (GML). Il comando GML è un carattere singolo in una stringa eventualmente seguito da uno o più argomenti.

L'istruzione DRAW è valida soltanto in modalità grafica.

Comandi di spostamento

I comandi di spostamento che seguono iniziano lo spostamento dalla posizione grafica corrente. Di solito questa è la coordinata dell'ultimo punto grafico tracciato con un altro comando GML, LINE o PSET. Quando il programma viene eseguito, la posizione corrente predefinita è al centro dello schermo (160,100 in media risoluzione; 320,100 in alta risoluzione). I comandi di spostamento eseguono lo spostamento per la distanza specificata dalla dimensione del fattore ***n***, il cui valore predefinito è 1; per cui, se viene omesso il fattore ***n*** e viene usata la dimensione predefinita, i comandi effettuano lo spostamento di un punto.

Comando	*Spostamenti*
U*n*	su
D*n*	giù
L*n*	a sinistra
R*n*	a destra
E*n*	diagonalmente in alto a destra
F*n*	diagonalmente in basso a destra
G*n*	diagonalmente in basso a sinistra
H*n*	diagonalmente in alto a sinistra

Questo comando esegue spostamenti sulla base di quanto indicato nei seguenti argomenti:

M*x*,*y* — Spostamento assoluto o relativo. Se *x* è preceduto da un segno di addizione (+) o di sottrazione (-), *x* e *y* vengono aggiunte alla posizione grafica corrente, e collegate a quella posizione tramite una riga. Altrimenti, viene disegnata una riga al punto *x*,*y*, dalla posizione corrente.

I comandi che seguono possono precedere ognuno dei comandi di spostamento sopra elencati:

B — Sposta senza tracciare punti.

N — Sposta e, a spostamento effettuato, ritorna alla posizione d'origine.

Sono anche disponibili i comandi seguenti:

A*n* — Imposta l'angolo *n*. *n* può variare da 0 a 3, dove 0 è 0°, 1 è 90°, 2 è 180° e 3 è 270°. Le figure ruotate di 90° o 270° vengono riprodotte in scala in modo da apparire sullo schermo con la stessa dimensione di figure a 0° o a 180° su un monitor con la proporzione predefinita di 4:3.

TA*n* — Ruota l'angolo di *n*. *n* può essere qualsiasi valore tra -360 e +360. Se il valore specificato da *n* è positivo, l'angolo viene ruotato in senso antiorario. Se invece il valore specificato da *n* è negativo, l'angolo viene ruotato in senso orario.

Comando	***Spostamenti***
C*n*	Imposta il colore *n*. Per una discussione più dettagliata riguardante i colori validi, i numeri e gli attributi, si vedano le istruzioni COLOR, PALETTE e SCREEN.
S*n*	Imposta il fattore di scala. *n* può variare da 1 a 255 e deve essere diviso per 4 per derivarne il fattore di scala. Il fattore di scala viene poi moltiplicato per le distanze fornite da D, L, R, E, F, G, H, o dai relativi comandi M per ottenere le distanze effettive. Il valore predefinito di S è 4.
x*stringa;variabile*	Esegue sottostringhe. Questo comando esegue una seconda sottostringa da una stringa, in modo simile a GOSUB. Una stringa esegue un'altra, la quale esegue una terza stringa, e così via. *stringa* rappresenta una variabile assegnata ad una stringa di comandi di spostamento.
P*colore,bordo*	Specifica i colori per le figure grafiche e crea una figura piena. *colore* specifica il colore con il quale riempire la figura. *bordo* specifica il colore del bordo (contorno). Per maggiori dettagli riguardanti i colori validi, i numeri e gli attributi, si vedano le istruzioni COLOR, PALETTE, e SCREEN. E' necessario specificare valori sia per il *colore* che per il *bordo*. Questo comando (P*colore,bordo*) non colora i motivi.

Argomenti numerici

Gli argomenti numerici possono essere costanti del tipo "123" o "=*variabile*;", in cui *variabile* rappresenta il nome di una variabile.

Quando si usa la seconda sintassi, "=*variabile*;", l'inclusione del punto e virgola è necessario. Altrimenti, l'inclusione del punto è virgola tra comandi è facoltativo.

Le variabili possono anche essere specificate usando VARPTR$(*variabile*).

Esempio 1

Per disegnare un quadrato in risoluzione media:

```
10 SCREEN 1
20 A=20
30 DRAW "U=A;R=A;D=A;L=A;"
RUN
```

Esempio 2

La proporzione di esposizione per disegnare un quadrato su uno schermo standard è 4:3, come illustrato sotto.

Per disegnare un quadrato largo 96 pixel su uno schermo di 640 X 200 pixel (SCREEN 2), eseguire i calcoli seguenti:

```
Valore orizzontale = 96
Valore verticale = 96*(200/640)*(4/3)
```

oppure

```
Valore verticale = 40
Valore orizzontale = 40*(640/200)*(3/4)
```

I valori orizzontali sono 4/3 dei valori verticali.

Esempio 3

Per disegnare un triangolo in risoluzione media:

```
10 CLS
20 SCREEN 1
30 PSET (60,125)
40 DRAW "E100; F100; L199"
RUN
```

EDIT (comando)

Funzione

Visualizza la riga specificata e porta il cursore sotto la prima cifra del numero di riga, in modo da permetterne la modifica.

Sintassi

EDIT *numeroriga*
EDIT .

Commenti

numeroriga rappresenta il numero di una riga esistente nel programma.

Il punto (.) indica la riga corrente. Il comando seguente permette la modifica della riga corrente:

```
EDIT .
```

Quando una riga viene inserita, diventa automaticamente la riga corrente.

La riga corrente è sempre l'ultima riga a cui hanno fatto riferimento l'istruzione EDIT, il comando LIST o un messaggio d'errore.

Se *numeroriga* è una riga non esistente nel programma, si produce l'errore `Numero di riga non definito.`

Esempi

```
EDIT 150
```

Visualizza la riga numero 150 per la modifica.

END (istruzione)

Funzione

Termina l'esecuzione del programma, chiude i file e ritorna al livello di comando.

Sintassi

END

Commenti

L'istruzione END può essere posta in qualsiasi punto del programma per terminarne l'esecuzione.

Diversamente dall'istruzione STOP, END non provoca la riproduzione del messaggio `Interruzione alla riga xxxx`.

L'inserimento dell'istruzione END alla fine del programma è facoltativa. Dopo l'esecuzione di END, GW-BASIC ritorna sempre al livello di comando.

END chiude tutti i file.

Esempi

```
520 IF K>1000 THEN END ELSE GOTO 20
```

Se il valore di K eccede 1000 il programma viene chiuso e si ritorna al livello di comando.

ENVIRON (istruzione)

Funzione

Permette all'utente la modifica di parametri nella tabella di stringhe d'ambiente di GW-BASIC. Può essere usato per cambiare il percorso di un parametro per operare su una directory figlia (si vedano ENVIRON$, SHELL ed il comando PATH di MS-DOS) o per passare dei parametri ad una directory figlia inventando un parametro di ambiente nuovo.

Sintassi

ENVIRON *stringa*

Commenti

stringa rappresenta l'espressione alfanumerica valida contenente il parametro della nuova stringa d'ambiente.

stringa deve essere impostato nella forma seguente

nomeparametro=testo

dove *nomeparametro* rappresenta il nome del parametro (ad esempio PATH).

nomeparametro deve essere separato dal testo con un segno di uguaglianza (=) o con uno spazio. ENVIRON considera tutto quanto viene posto sulla sinistra del primo spazio o del segno di uguaglianza (=) come *nomeparametro* e tutto quello che segue come testo.

testo rappresenta il nuovo testo del parametro. Se *testo* è una stringa nulla, o contiene soltanto un punto e virgola, la *stringa* (compreso *nomeparametro* =) viene rimossa dalla tabella di stringhe d'ambiente e la tabella viene compressa. *Testo* non può contenere alcuno spazio vuoto intermedio.

Se *nomeparametro* non esiste, la *stringa* viene aggiunta alla fine della tabella di stringhe d'ambiente.

Se invece *nomeparametro* esiste, viene cancellato, la tabella di stringhe d'ambiente viene compressa e la nuova stringa viene aggiunta alla fine della tabella.

Esempi

L'istruzione seguente crea un percorso predefinito verso la directory principale sul disco A, nell'ipotesi che la tabella di stringhe dell'ambiente sia vuota.

```
ENVIRON "PATH=A:\"
```

Anche se la subdirectory di lavoro fosse, ad esempio, \UTENTE\FRANCA, sarebbe possibile ottenere DEBUG dalla directory principale.
Si può anche aggiungere un nuovo parametro:

```
ENVIRON "COMSPEC=A:\COMMAND.COM"
```

A questo punto la tabella di stringhe d'ambiente contiene:

```
PATH=A:\;COMSPEC=A:\COMMAND.COM
```

Il percorso può essere cambiato in un nuovo valore:

```
ENVIRON "PATH=A:\FATTURE;A:\CLIENTI"
```

Utilizzando la funzione ENVIRON$ insieme all'istruzione ENVIRON si può aggiungere il parametro del percorso:

```
ENVIRON "PATH="+ENVIRON$("PATH")+";B:\CAMPIONI"
```

Per finire, si cancella COMSPEC:

```
ENVIRON "COMSPEC=;"
```

A questo punto la tabella di stringhe dell'ambiente contiene

```
PATH=A:\FATTURE;A:\CLIENTI;B:\CAMPIONI
```

ENVIRON$ (funzione)

Funzione

Permette all'utente di estrarre una determinata stringa d'ambiente dalla tabella d'ambiente.

Sintassi

v$=ENVIRON$(*nomeparametro***)**
v$=ENVIRON$(*nparam***)**

Commenti

nomeparametro rappresenta un'espressione alfanumerica valida contenente il parametro da ricercare.

nparam rappresenta un numero intero nell'intervallo numerico che va da 1 a 255.

Se viene usato un argomento alfanumerico, ENVIRON$ restituisce, dalla tabella di stringhe d'ambiente, una stringa contenente il testo che segue *nomeparametro=*.

Se *nomeparametro* non viene trovato, viene restituita una stringa nulla.

Se viene usato un argomento numerico (*nparam*), ENVIRON$ restituisce una stringa contenente l'*n*-esimo parametro della tabella di stringhe d'ambiente.

Se non esiste alcun parametro *nparam*, viene restituita una stringa nulla.

La funzione ENVIRON$ fa distinzione tra lettere maiuscole e minuscole.

Esempi

Le righe seguenti:

```
ENVIRON "PATH=A:/FATTURE;A:/CLIENTI;B:/BANCHE"
        'Crea la voce
PRINT ENVIRON$("PATH") 'Stampa la voce
```

determinano la stampa delle stringhe seguenti:

```
A:/FATTURE;A:/CLIENTI;B:/BANCHE
```

La riga seguente stampa la prima stringa nell'ambiente:

```
PRINT ENVIRON$(1)
```

Il programma seguente salva la tabella di stringhe d'ambiente in una matrice in modo da permettere la modifica della figlia. Conseguentemente al completamento dell'operazione sulla figlia, l'ambiente viene ripristinato.

```
10    DIM AMBTBL$(10)"
20    NPARMS= 1
30    WHILE LEN(ENVIRON$(NPARS)) >0
40    AMBTBL$ (NPARMS)= ENVIRON$(NPARMS)
50    NPARMS= NPARMS + 1
60    WEND
70    NPARMS= NPARMS-1
72    WHILE LEN(ENVIRON$(1))>0
73    A$=MID$(ENVIRON$(1),1,INSTR (ENVIRON$(1),"="))
74    ENVIRON A$+";"
75    WEND
90    ENVIRON "MIOFIGPARM1=ORDINA PER NOME"
100   ENVIRON "MIOFIGPARM2=ELENCA PER NOME"
      .
      .
      .
1000 SHELL "MIOFIG"'RUNS "MIOFIG.EXE"
1002 WHILE LEN(ENVIRON$(1))>0
1003 A$=MID$(ENVIRON$(1),1,INSTR(ENVIRON$ (1),"="00
1004 ENVIRON A$+";"
1005 WEND
1010 FOR I=1 TO NPARMS
1020 ENVIRON ENVTBL$(I)
1030 NEXT I
      .
      .
      .
```

L'istruzione DIM alla riga 10 ipotizza di non accedere a più di 10 parametri.

Alla riga 20, il numero iniziale dei parametri viene impostato ad 1.

Da riga 30 a 70 si utilizzano una serie di istruzioni che hanno la funzione di modificare e correggere i numeri di parametro.

La riga 71 cancella l'ambiente attuale.

La riga 74 cancella la stringa.

Le righe da 90 a 100 memorizzano l'ambiente nuovo.

Le righe da 1000 a 1030 ripetono la procedura cancellando l'ambiente attuale e ripristinando i parametri impostati nella prima parte del programma.

EOF (funzione)

Funzione

Restituisce -1 (reale) quando si giunge al termine di un file sequenziale o di un file di comunicazioni, oppure restituisce 0 se la fine del file (EOF) non viene raggiunta.

Sintassi

v=EOF*(numerofile)*

Commenti

Se un comando GET viene eseguito dopo la fine del file, EOF restituisce -1. Ciò può essere usato per conoscere la dimensione del file utilizzando una ricerca binaria o un altro algoritmo. Con i file di comunicazioni, -1 indica che il buffer è vuoto.

Usare EOF per verificare la fine del file ed evitare gli errori del tipo `INPUT dopo la fine` nella fase di lettura dati.

Esempi

```
10 OPEN "I",1,"DATI"
20 C=0
30 IF EOF(1) THEN 100
40 INPUT#1,M(C)
50 C=C+1:GOTO 30
100 END
RUN
```

Il file DATI viene letto nella matrice M fino al raggiungimento del termine del file, dopodiché il programma si sposta alla riga 100.

ERASE (istruzione)

Funzione

Elimina matrici da un programma.

Sintassi

ERASE *elenco variabili di matrice*

Commenti

Conseguentemente alla loro cancellazione, le matrici possono essere ridimensionate, oppure lo spazio di memoria loro assegnato precedentemente può essere utilizzato per altri scopi.

Se si tenta di ridimensionare una matrice senza prima cancellarla, si produce un errore.

Esempi

```
200 DIM B (250)
.
.
.
450 ERASE A,B
460 DIM B(3,4)
```

Le matrici A e B vengono eliminate dal programma. La matrice B viene ridimensionata a 3 colonne per 4 righe, cioè 12 elementi, tutti impostati al valore 0.

ERDEV e ERDEV$ (variabili)

Funzione

Restituisce il valore effettivo (ERDEV) di un errore periferico insieme al nome dell'unità (ERDEV$) che lo ha causato.

Sintassi

ERDEV
ERDEV$

Commenti

ERDEV contiene il codice dell'errore dall'interruzione 24H negli 8 bit inferiori. I bit da 8 a 15 dall'attributo del blocco di intestazione della periferica (DHB) vengono elencati direttamente negli 8 bit superiori.

Se l'errore si trova in una periferica a caratteri, ERDEV$ conterrà il nome (8 byte) della periferica a caratteri. Se la periferica non è a caratteri, ERDEV$ conterrà il nome (2 byte) della periferica a blocchi (A:, B:, ecc.).

Esempi

Il file di descrizione della periferica installata lpt2: ha provocato l'errore `Senza carta` via INT 24H.

Negli 8 bit inferiori, ERDEV contiene l'errore numero 9, mentre in quelli superiori contiene il byte superiore degli attributi dell'intestazione della periferica.

```
ERDEV$ contiene "LPT2:   ".
```

ERR ed ERL (variabili)

Funzione

Restituisce il codice d'errore (ERR) e il numero di riga (ERL) associato all'errore.

Sintassi

v=ERR
v=ERL

Commenti

La variabile ERR contiene il codice d'errore dell'ultimo errore verificatosi. L'appendice A del *Manuale dell'utente* contiene un elenco di tutti i codici d'errore con le rispettive definizioni.

La variabile ERL contiene il numero della riga nella quale si è verificato l'errore.

Normalmente, le variabili ERR e ERL vengono usate in combinazione con le istruzioni IF-THEN, ON ERROR...GOTO o GOSUB per dirigere il flusso del programma verso il trapping degli errori.

Se l'istruzione che ha causato l'errore è in modalità diretta, ERL conterrà 65535. Per verificare se un errore è stato causato da un'istruzione in modalità diretta, usare una riga del tipo seguente:

```
IF 65535=ERL THEN...
```

Altrimenti, usare le seguenti istruzioni:

```
10 IF ERR=codice errore THEN...GOSUB 4000
20 IF ERL=numero riga THEN...GOSUB 4010
```

Avvertenza Se il numero di riga non si trova sulla parte destra dell'operatore relazionale, la rinumerazione tramite RENUM non può essere effettuata.

Essendo le variabili ERL ed ERR riservate, non possono mai apparire nella parte sinistra del segno di uguaglianza (=) nell'istruzione LET.

ERROR (istruzione)

Funzione

Simula un errore, o permette all'utente di definire codici d'errore.

Sintassi

ERROR *espressione intera*

Commenti

Il valore di *espressione intera* deve essere superiore a 0 ed inferiore a 255.

Se il valore di *espressione intera* è uguale ad un codice d'errore già in uso, l'istruzione ERROR simula il verificarsi di quell'errore, e stampa poi il messaggio d'errore corrispondente.

Un codice d'errore definito dall'utente deve utilizzare un valore superiore a qualsiasi valore usato dai codici d'errore di GW-BASIC. Attualmente i codici d'errore di GW-BASIC sono 76. Comunque, a causa del probabile incremento di tale numero, è preferibile usare un numero di codice alto abbastanza da rimanere valido anche in futuro.

L'uso dei codici d'errore definiti dall'utente è permesso in routine per il trapping di errori.

Se un'istruzione ERROR specifica un codice per il quale non è stato definito un messaggio d'errore, GW-BASIC emette il messaggio `Errore non riproducibile`.

L'esecuzione di un'istruzione ERROR per la quale non esiste una routine di trapping di errori, provoca l'emissione di un messaggio d'errore e l'interruzione dell'esecuzione stessa.

Per un elenco completo di codici e messaggi d'errore già definiti in GW-BASIC, consultare l'appendice A nel *Manuale dell'utente*.

Esempi

Gli esempi seguenti simulano l'errore 15 (il codice rappresentante `Stringa troppo lunga`):

```
LIST
10 S=10
20 T=5
30 ERROR S+T
40 END
Ok
RUN
```

Oppure in modalità diretta:

```
Ok
ERROR 15       (riga digitata dall'utente)
Stringa troppo lunga      (riga digitata da GW-BASIC)
Ok
```

L'esempio seguente comprende un messaggio d'errore definito dall'utente:

```
.
.
.
110 ON ERROR GOTO 400
120 INPUT "Qual'è la somma?";B
130 IF B>5000 THEN ERROR 210
.
.
.
400 IF ERR=210 THEN PRINT "LIMITE MASSIMO 5000"
410 IF ERL=130 THEN RESUME 120
.
.
.
```

EXP (funzione)

Funzione

Restituisce *e* (la base dei logaritmi naturali) elevato ad *x*.

Sintassi

EXP(*x*)

Commenti

x deve essere inferiore a 88.02969.

Se EXP eccede tale limite, viene visualizzato il messaggio d'errore `Valore troppo elevato`; viene allora fornito come risultato il valore infinito di macchina con il simbolo appropriato e l'esecuzione riprende.

Il calcolo di EXP avviene in precisione singola, a meno che il parametro **/d** sia stato usato all'esecuzione di GW-BASIC.

Esempi

```
10 X = 5
20 PRINT EXP(X-1)
RUN
        54.59815
Ok
```

Stampa il valore di *e* elevato alla quarta.

EXTERR (funzione)

Funzione

Restituisce l'informazione d'errore esteso.

Sintassi

EXTERR(*n*)

Commenti

EXTERR restituisce l'informazione d'errore estesa fornita dalle versioni di DOS 3.0 e maggiori. Per le versioni di DOS precedenti a 3.0, EXTERR restituisce sempre 0. Il singolo argomento intero deve rientrare nell'intervallo numerico che va da 0 a 3 in questo modo:

Valore di n	*Valore restituito*
0	Codice dell'errore esteso
1	Classe dell'errore esteso
2	Azione suggerita dall'errore esteso
3	Posizione dell'errore esteso

I valori restituiti non sono definiti da GW-BASIC, bensì da DOS.

Il codice d'errore esteso viene caricato e salvato da GW-BASIC ogni volta che una funzione appropriata di DOS viene eseguita. Per cui, quando viene effettuata la chiamata di una funzione EXTERR, questi valori salvati vengono restituiti.

FIELD (istruzione)

Funzione

Assegna spazio per le variabili in un buffer di file ad accesso casuale.

Sintassi

FIELD [>] *numerofile*, *larghezza* **AS** *variabilestringa* [,*larghezza* **AS** *variabilestringa*]...

Commenti

numerofile rappresenta il numero sotto il quale è avvenuta l'apertura del file.

larghezza rappresenta il numero di caratteri da assegnare alla variabile alfanumerica.

variabilestringa rappresenta la variabile alfanumerica da utilizzare per l'accesso casuale al file.

L'istruzione FIELD deve già essere stata eseguita prima di poter:

- ottenere i dati da un buffer casuale dopo l'istruzione GET
- inserire dei dati prima dell'istruzione PUT

Ad esempio, la riga seguente assegna le prime 20 posizioni (byte) di un buffer di file ad accesso casuale alla variabile alfanumerica N$, le 10 posizioni seguenti ad ID$, e le 40 posizioni che seguono ad ADD$:

```
FIELD 1, 20 AS N$, 10 AS ID$, 40 AS ADD$
```

La funzione di FIELD è di assegnare soltanto spazio; FIELD non inserisce dati nel buffer del file ad accesso casuale.

Il numero totale dei byte assegnati da FIELD non deve eccedere la lunghezza del record specificata all'apertura del file. Altrimenti, si produce l'errore `FIELD di dimensioni eccessive` (la lunghezza predefinita del record è 128).

Nello stesso file è possibile eseguire un numero indeterminato di istruzioni FIELD, e tutte le istruzioni FIELD vengono eseguite simultaneamente.

Avvertenza Non utilizzare il nome di variabile il cui spazio è stato impostato con FIELD, in un'istruzione INPUT o LET. Una volta assegnato lo spazio, il nome di variabile punta in direzione della posizione corretta nel buffer di file ad accesso casuale. Se in seguito si eseguono INPUT o LET con quel nome di variabile, il puntatore della variabile viene spostato in direzione dello spazio di stringa (si vedano le istruzioni LSET/RSET e GET).

FILES (comando)

Funzione

Stampa i nomi dei file nell'unità disco specificata.

Sintassi

FILES [*nomepercorso*]

Commenti

Se il *nomepercorso* viene omesso, il comando elenca tutti i file contenuti nella directory corrente dell'unità disco scelta. *Nomepercorso* può contenere dei punti interrogativi (?) ad indicare qualsiasi carattere nel nome del file o nell'estensione. Un asterisco (*) usato come primo carattere del nome del file o dell'estensione indica qualsiasi file o estensione.

Inoltre, nella sintassi sono presenti anche il nome della directory ed il numero dei byte nel file. Quando viene usata una directory con struttura ad albero, vengono anche visualizzati due simboli speciali.

Le subdirectory sono contrassegnate da *<DIR>* dopo il nome della directory.

Esempi

```
FILES
FILES "*.BAS"
FILES "B:*.*"
FILES "TEST?.BAS"
```

Il comando FILES permette l'uso di nomi di percorso. La directory del percorso specificato viene visualizzata. Se il percorso non viene specificato, la directory corrente viene usata come valore predefinito.

```
FILES "CLIENTI\"
```

Per elencare tutti i file con estensione .BER nella directory chiamata CLIENTI sul dischetto nell'unità disco B:

```
FILES "B:CLIENTI\*.BER"
```

FIX (funzione)

Funzione

Riduce *x* ad un numero intero.

Sintassi

FIX(*x*)

Commenti

La funzione FIX non esegue l'arrotondamento dei numeri, ma elimina semplicemente tutte le cifre alla destra del punto decimale.

FIX(*x*) equivale a SGN(*x*)*INT(ABS(*x*)). La differenza principale esistente tra FIX ed INT è che FIX non restituisce il numero seguente più basso per *x* negativo.

FIX è utile per i moduli aritmetici.

Esempi

```
PRINT FIX(58.75)
        58
Ok
PRINT FIX(-58.75)
        -58
Ok
```

FOR e NEXT (istruzioni)

Funzione

Eseguono un ciclo d'istruzioni un determinato numero di volte.

Sintassi

FOR *variabile=x* **TO** *y* **[STEP** *z***]**
.
.
.
NEXT [*variabile*][,*variabile...*]

Commenti

La *variabile* viene usata come "contatore".

x,y e *z* sono espressioni numeriche.

STEP *z* specifica l'incremento del contatore per ciascun ciclo di istruzioni.

La prima espressione numerica (*x*) rappresenta il valore iniziale del contatore. La seconda espressione numerica (*y*) rappresenta il valore finale del contatore.

Le righe del programma che seguono l'istruzione FOR vengono eseguite fino a che si incontra l'istruzione NEXT. Dopodiché il contatore viene incrementato del valore specificato da STEP.

Se STEP non viene specificato, il contatore viene incrementato di 1.

Il valore del contatore viene controllato per vedere se è maggiore del valore finale (*y*). Se non lo è, GW-BASIC si sposta all'istruzione che segue immediatamente FOR, e ripete il ciclo. In caso contrario, l'esecuzione continua con l'istruzione che segue l'istruzione NEXT. Questo è appunto detto ciclo di istruzioni FOR NEXT.

Se il valore iniziale del ciclo è superiore al valore finale, il ciclo non viene eseguito.

Se STEP è negativo, il valore finale del contatore viene impostato ad un valore inferiore al valore iniziale. Il valore del contatore viene diminuito conseguentemente a ciascuna esecuzione del ciclo, ed il ciclo viene eseguito fino a che il valore del contatore è inferiore al valore finale.

Cicli d'istruzioni nidificati

I cicli FOR NEXT possono essere nidificati; Un ciclo FOR NEXT può essere cioè posto all'interno di un altro ciclo FOR NEXT. In presenza di una nidificazione dei cicli, ciascun ciclo deve avere un nome di variabile (contatore) differente.

L'istruzione NEXT del ciclo interno deve apparire prima di quella del ciclo esterno.

Se più cicli nidificati hanno lo stesso punto finale, si può usare una sola istruzione NEXT per tutti.

La *variabile* nell'istruzione NEXT può essere omessa, a significare che NEXT corrisponde all'istruzione FOR più recente.

Se l'istruzione NEXT viene incontrata prima della corrispondente istruzione FOR, viene visualizzato il messaggio d'errore `NEXT senza FOR` e l'esecuzione si arresta.

Esempi

L'esempio che segue stampa i valori interi della variabile I% da 1 a 10 con incremento 2. Per un'esecuzione più rapida, I è dichiarata come valore intero attraverso il segno %.

```
10 K=10
20 FOR I%=1 TO K STEP 2
30 PRINT I%
.
.
.
60 NEXT
RUN
 1
 3
 5
 7
 9
Ok
```

Nell'esempio seguente, il ciclo non viene eseguito, in quanto il valore iniziale del ciclo eccede il valore finale. In questo caso non viene stampato nulla.

```
10 R=0
20 FOR S=1 TO R
30 PRINT S
40 NEXT S
```

Nell'esempio che segue, il ciclo viene eseguito 10 volte. Il valore finale della variabile del ciclo viene sempre impostato prima del valore iniziale.

```
10 S=5
20 FOR S=1 TO S+5
30 PRINT S;
40 NEXT
RUN
      1  2  3  4  5  6  7  8  9  10
Ok
```

FRE (funzione)

Funzione

Restituisce il numero di byte disponibili nella memoria di stringa assegnata.

Sintassi

FRE(*x*$)
FRE(*x*)

Commenti

Gli argomenti (*x*$) e (*x*) sono argomenti fittizi.

Prima che FRE (*x*$) restituisca il valore dello spazio disponibile nella memoria della stringa assegnata, GW-BASIC inizia un'attività di "recupero rifiuti". I dati contenuti nello spazio di memoria di stringa vengono raccolti e riorganizzati, e le porzioni inutilizzate delle stringhe frammentate vengono cancellate per fare spazio a nuovi dati.

Se la funzione FRE non viene usata, GW-BASIC inizia l'attività di "recupero rifiuti" non appena lo spazio di memoria di stringa si esaurisce. GW-BASIC non inizia però tale operazione prima che tutta la memoria libera sia stata utilizzata. Il "recupero rifiuti" può durare da 1 a 1,5 minuti.

FRE("") o con qualsiasi altra stringa impone un "recupero rifiuti" prima dell'indicazione del numero dei byte disponibili. Per cui, l'uso periodico di FRE("") comporta minori ritardi in occasione di ogni operazione di "recupero rifiuti".

Si noti che la funzione CTRL-BREAK non può essere utilizzata durante questa operazione di recupero di spazio in memoria.

Esempi

```
PRINT FRE(0)
         14542
Ok
```

Nota Il valore restituito può essere diverso.

GET (File) (istruzione)

Funzione

Legge un record da un file di disco ad accesso casuale in un buffer ad accesso casuale.

Sintassi

GET [>]*numerofile*[,*numerorecord*]

Commenti

numerofile rappresenta il numero sotto il quale è stata eseguita l'apertura del file.

numerorecord rappresenta il numero del record, e rientra nell'intervallo numerico che va da 1 a 16.777.215.

Se *numerorecord* viene omesso, viene letto nel buffer il primo record successivo all'ultimo GET.

Dopo GET, si possono usare INPUT> e LINE INPUT> per leggere i caratteri dal buffer del file ad accesso casuale.

GET può anche essere usato per i file di comunicazioni. *numerorecord* rappresenta il numero dei byte da leggere dal buffer di comunicazioni. *numerorecord* non può eccedere la lunghezza del buffer impostata con l'istruzione OPEN COM(*n*).

Esempi

L'esempio che segue apre il file VENDITE per l'accesso casuale, definisce i campi, legge un record e lo visualizza:

```
10 OPEN "R",1,"A:VENDITE.FIL"
20 FIELD 1,30 AS VENDNOMI$,20 AS IND$,15 AS CITTA$
30 GET 1
40 PRINT VENDNOMI$,IND$,CITTA$
50 CLOSE 1
```

Questo esempio apre il file VENDITE.FIL per l'accesso casuale, con i campi definiti alla riga 20. Alla riga 30, l'istruzione GET esegue la lettura di un record nel buffer del file. Alla riga 40 vengono visualizzate le informazioni provenienti dal record appena letto. La riga 50 chiude il file.

GET (Grafici) (istruzione)

Funzione

Trasferisce le immagini grafiche dallo schermo.

Sintassi

GET (*x1,y1*)-(*x2,y2*), *nomematrice*

Commenti

Le istruzione PUT e GET vengono usate per il trasferimento delle immagini grafiche dallo e sullo schermo. PUT e GET rendono possibile l'animazione ed il movimento dell'oggetto ad alta velocità in qualunque modalità grafica.

L'istruzione GET trasferisce l'immagine dello schermo delimitata dal rettangolo descritto dai punti specificati nella matrice. Il rettangolo viene definito allo stesso modo in cui viene definito il rettangolo tracciato dall'istruzione LINE, usando l'opzione **,B**.

La matrice è semplicemente il luogo contenente l'immagine; essa può essere di qualsiasi genere eccetto che una stringa. La matrice deve essere impostata ad una dimensione larga abbastanza da contenere l'intera immagine. Se interpretati direttamente, i contenuti della matrice dopo l'esecuzione di GET sono privi di significato (a meno che la matrice sia di tipo intero, come mostrato sotto).

Il formato della memoria nella matrice è impostato nel modo seguente:

- 2 byte con dimensione di x pixel
- 2 byte con dimensione di y pixel
- la matrice di dati

I dati per ciascuna riga di pixel sono allineati a sinistra al livello di un confine di byte. Se vengono memorizzati meno di 8 bit, il resto del byte viene riempito di zeri. La dimensione in byte necessaria per la matrice è la seguente:

4+INT((x**bitperpixel*+7)/8)*y

Per i valori di bit per pixel per modalità di schermo diverse, vedere l'istruzione SCREEN.

I byte per elemento di una matrice sono i seguenti:

- 2 per interi
- 4 in precisione singola
- 8 in precisione doppia

Il numero di byte necessario per ottenere un'immagine di 10 per 12 in una matrice a valori interi è 4+INT((10*2+7)/8)*12, o 40 byte. E' necessario che una matrice intera abbia almeno 20 elementi.

Se OPTION BASE è uguale a zero, si può utilizzare una matrice intera per esaminare le dimensioni di x e y e dei dati. La dimensione di x si trova nell'elemento 0 della matrice, mentre la dimensione di y si trova nell'elemento 1. I valori interi vengono memorizzati prima con il byte inferiore e poi con quello superiore, mentre i dati vengono trasferiti prima con il byte superiore e poi con quello inferiore.

E' possibile ottenere un'immagine in una modalità e trasferirla in un'altra, anche se l'effetto che si ottiene può risultare strano per via del modo in cui i punti vengono rappresentati in ciascuna modalità.

Esempi

```
10  CLS:SCREEN 1
20  PSET (130,120)
30  DRAW "U25;E7;R20;D32;L6;U12;L14"
40  DRAW "D12;L6":PSET(137,102)
50  DRAW "U4;E4;R8;D8;L12
60  PSET(137,88)
70  DRAW "U4;R20;D32;G4":PAINT(139,87)
80  DIM A(500)
90  GET (125,130) - (170,80),A
100 FOR I=1 TO 1000:NEXT I
110 PUT (20,20),A,PSET
120 FOR I=1 TO 1000:NEXT I
130 GET (125,130) - (170,80),A
140 FOR I=1 TO 1000:NEXT I
150 PUT (220,130),A,PRESET
```

GOSUB...RETURN (istruzione)

Funzione

Si collega a una subroutine per poi ritornare al punto di partenza.

Sintassi

GOSUB *numeroriga*
.
.
.
RETURN [*numeroriga*]

Commenti

numeroriga è il numero della prima riga della subroutine.

In un programma una subroutine può essere chiamata più volte; inoltre, una subroutine può essere chiamata dall'interno di un'altra subroutine. L'unico limite a questo tipo di nidificazione è la memoria disponibile.

Un'istruzione RETURN in una subroutine determina il ritorno di GW-BASIC all'istruzione che segue l'istruzione GOSUB più recente. Se la logica lo richiede, una subroutine può contenere anche più istruzioni RETURN.

Le subroutine possono apparire in qualsiasi punto del programma. Esse, però, devono poter essere facilmente distinte dal programma principale.

Per prevenire l'inclusione involontaria di istruzioni, far precedere la subroutine dall'istruzione STOP, END o GOTO, in modo da trasferire il controllo del programma alla subroutine.

Esempi

```
10 GOSUB 40
20 PRINT "RITORNO DALLA SUBROUTINE"
30 END
40 PRINT "SUBROUTINE";
50 PRINT " IN";
60 PRINT " CORSO"
70 RETURN
RUN
SUBROUTINE IN CORSO
RITORNO DALLA SUBROUTINE
Ok
```

L'istruzione END alla riga 30 impedisce la riesecuzione della subroutine.

GOTO (istruzione)

Funzione

Si ricollega incondizionatamente dal flusso normale di esecuzione al numero di riga specificato.

Sintassi

GOTO *numeroriga*

Commenti

numeroriga è qualsiasi numero di riga valido nel programma.

Se *numeroriga* contiene un'istruzione eseguibile, quell'istruzione e quelle che la seguono vengono eseguite. In caso contrario, l'esecuzione procede con la prima istruzione incontrata dopo *numeroriga*.

Esempi

```
10 READ R
20 PRINT "R =";R;
30 A = 3.14*R^2
40 PRINT "AREA =";A
50 GOTO 10
60 DATA 5, 7, 12
RUN
R = 5   AREA = 78.5
R = 7   AREA = 153.86
R = 12  AREA = 452.16
DATA non presente alla riga 10
Ok
```

Il messaggio viene generato quando il programma tenta di leggere la quarta istruzione DATA (che non esiste) alla riga 60.

HEX$ (funzione)

Funzione

Restituisce una stringa rappresentante il valore esadecimale dell'argomento numerico.

Sintassi

***v*$=HEX$(x)**

Commenti

HEX$ converte i valori decimali che rientrano nell'intervallo da -32767 a +32768 in un'espressione alfanumerica esadecimale che rientra nell'intervallo esadecimale da 0 a FFFF.

I numeri esadecimali hanno come base 16, anzichè 10 (numeri decimali). Le appendici C e G nel *Manuale dell'utente* contengono informazioni più dettagliate circa i numeri esadecimali e i rispettivi equivalenti.

Prima che HEX$(x) venga calcolato, x viene arrotondato ad un numero intero.

Se x è negativo, viene usato il complemento binario di 2.

Esempi

```
10 CLS:INPUT "INSERIRE NUMERO DECIMALE";X
20 A$=HEX$(X)
30 PRINT X "DECIMALE E' "A$" ESADECIMALE"
RUN
INSERIRE NUMERO DECIMALE? 32
           32 DECIMALE E' 20 ESADECIMALE
Ok
```

IF (istruzione)

Funsione

Opera una scelta riguardante il flusso del programma sulla base del risultato restituito dall'espressione.

Sintassi

IF *espressione*[,] **THEN** *istruzione(i)*[,][**ELSE** *istruzione(i)*]
IF *espressione*[,] **GOTO** *numeroriga*[[,] **ELSE** *istruzione(i)]*

Commenti

Se il risultato dell'espressione non è zero, viene eseguita la riga di THEN o GOTO.

Se invece il risultato dell'*espressione* è zero (falso), l'istruzione di THEN o GOTO viene ignorata ed il numero di riga di ELSE, se indicata, viene eseguita. Altrimenti, l'esecuzione prosegue per il normale flusso del programma. Una virgola va inserita prima di THEN ed ELSE.

Con THEN ed ELSE si può inserire un numero di riga a cui collegarsi, oppure una o più istruzioni da eseguire.

GOTO è sempre seguito da un numero di riga.

Se l'istruzione non contiene lo stesso numero di ELSE e THEN, ciascun ELSE viene abbinato al THEN non ancora abbinato più vicino. Ad esempio:

```
IF A=B THEN IF B=C THEN PRINT "A=C" ELSE PRINT "A < >
C"
```

questa riga non stampa "A<>C" quando A<>B.

Se l'istruzione IF...THEN è seguita da un numero di riga in modalità diretta, si produce l'errore `Numero di riga non definito`, a meno che un'istruzione con il numero di riga specificato non sia stata inserita precedentemente in modalità indiretta.

Siccome IF...THEN...ELSE viene usato come un'unica istruzione, ELSE non può essere inserito in una riga separata. L'intera istruzione deve essere in una sola riga.

La nidificazione delle istruzioni IF

Le istruzioni IF.. THEN...ELSE possono essere nidificate. L'unico limite alla nidificazione è la lunghezza della riga. Ad esempio, l'istruzione seguente è lecita:

```
100 IF X > Y THEN PRINT "MAGGIORE" ELSE IF Y > X THEN&
110   PRINT "MINORE"
200 ELSE PRINT "UGUALE"
```

Prova di uguaglianza

Quando si usa IF per verificare l'uguaglianza di un valore derivante dal risultato di un calcolo a punto mobile, ricordarsi che la rappresentazione interna del valore può essere non esatta. Per cui, è opportuno eseguire il test in rapporto all'intervallo di variazione del valore.

Ad esempio, per confrontare la variabile calcolata A in rapporto al valore 1.0, usare l'istruzione seguente:

```
100 IF ABS (A-1.0)<1.0E-6 THEN...
```

Questo test dà esito positivo se il valore di A è 1.0 con un errore relativo minore di 1.0E-6.

Esempi

L'istruzione seguente carica il record numero N, se N non è zero.

```
200 IF N THEN GET>1,N
```

L'esempio seguente contiene un test che determina se N è maggiore di 10 e minore di 20. Se N rientra in questo intervallo, viene calcolato DB per poi spostarsi alla riga 300. In caso contrario, l'esecuzione continua con la riga 110.

```
100 IF(N<20) and (N>10) THEN DB=1979-1:GOTO 300
110 PRINT "IL NUMERO NON RIENTRA NEI LIMITI"
```

L'istruzione seguente trasferisce dell'output al terminale o alla stampante di sistema, a secondo del valore di una variabile (IOFLAG). Se IOFLAG è zero, l'output viene trasferito alla stampante di sistema; altrimenti, viene trasferito al terminale.

```
210 IF IOFLAG THEN PRINT A$ ELSE LPRINT A$
```

INKEY$ (variabile)

Funzione

Restituisce un carattere letto dalla tastiera.

Sintassi

v$=INKEY$

Commenti

Se il buffer della tastiera non contiene alcun carattere in attesa, viene restituita una stringa nulla (di lunghezza zero).

Se vi sono vari caratteri in attesa, soltanto il primo viene restituito. La stringa sarà lunga uno o due caratteri.

Le stringhe a due caratteri vengono utilizzate per restituire i codici estesi descritti nell'appendice C del *Manuale dell'utente*. Il primo dei due caratteri del codice è zero.

Sullo schermo non viene visualizzato alcun carattere. Tutti i caratteri eccetto i seguenti vengono passati al programma:

CTRL-BREAK
CTRL-NUMLOCK
CTRL-ALT-DEL
CTRL-PRTSC
PRTSC

Esempi

```
10 CLS: PRINT"PREMERE RITORNO
20 TIMELIMIT% = 1000
30 GOSUB 1010
40 IF TIMEOUT% THEN PRINT "TROPPO TEMPO" ELSE PRINT
"BENE"
50 PRINT RISPOSTA$
60 END
.
.
.
1000 REM SUBROUTINE D'INPUT A TEMPO
1010 RESPONSE$=""
1020 FOR N%=1 TO TEMPOLIMITE%
1030 A$=INKEY$:IF LEN(A$)=0 THEN 1060
1040 IF ASC(A$)=13 THEN TIMEOUT%=0:RETURN
1050 RISPOSTA$=RISPOSTA$+A$
1060 NEXT N%
1070 TIMEOUT%=1 :RETURN
```

All'esecuzione di questo programma, se si preme il tasto RITORNO prima del completamento di 1000 cicli, il messaggio BENE viene stampato sullo schermo. Altrimenti, avviene la visualizzazione di TROPPO TEMPO.

Dal momento che l'istruzione INKEY$ percorre la tastiera una volta sola, deve essere posta tra cicli in modo da fornire all'operatore risposte in tempo adeguato.

INP (funzione)

Funzione

Restituisce il byte letto dalla porta della macchina *n*.

Sintassi

INP(*n*)

Commenti

n è un numero di porta di macchina valido che rientra nell'intervallo numerico che va da 0 a 65535.

La funzione INP permette la comunicazione di un dispositivo periferico con GW-BASIC.

INP è la funzione complementare all'istruzione OUT.

Esempi

```
100 A=INP(56)
```

All'esecuzione, la variabile A contiene il valore presente sulla porta 56. Il numero restituito rientra nell'intervallo da 0 a 255, decimale.

Il linguaggio di assemblaggio equivalente a questa istruzione è

```
MOV DX,56
IN AL,DX
```

INPUT (istruzione)

Funzione

Prepara il programma ad accettare input dal terminale durante l'esecuzione del programma stesso.

Sintassi

INPUT[;][*sollecito*;] *elencovariabili*
INPUT[;][*sollecito*,] *elencovariabili*

Commenti

sollecito è una stringa e rappresenta una richiesta di dati durante l'esecuzione del programma.

elencovariabili contiene la(e) variabile(i) che memorizza(no) i dati nella stringa di sollecito.

Ciascun dato contenuto in *sollecito* deve essere racchiuso tra virgolette, seguito da un punto e virgola o da una virgola e dal nome della variabile alla quale verrà assegnato. Se si indica più di una variabile, i gruppi di dati devono essere separati da virgole.

I dati inseriti vengono assegnati all'elenco di variabili. Il numero dei gruppi di dati forniti deve essere lo stesso delle variabili nell'elenco.

I nomi delle variabili nell'elenco possono essere numerici o alfanumerici Ciascun dato inserito deve essere coerente con il tipo specificato dal nome di variabile.

Un numero troppo alto o troppo basso di dati, od il tipo di valore errato (ad esempio, numerico anziché alfanumerico), provocano la visualizzazione del messaggio `?Iniziare da capo`. Finché non viene fornita una risposta accettabile, nessuna assegnazione di valori di input viene effettuata.

Dopo *sollecito* si può usare una virgola anzichè un punto e virgola, per eliminare il punto interrogativo. Ad esempio, la riga seguente stampa il sollecito senza il punto interrogativo:

```
INPUT        "INSERIRE DATA DI NASCITA",B$
```

Durante l'esecuzione del programma, i dati su quella riga vengono visualizzati, ed i dati dell'istruzione PRINT che segue vengono aggiunti alla riga.

Durante l'esecuzione, l'istruzione INPUT determina l'arresto del programma, seguito dalla visualizzazione della stringa di sollecito, per permettere all'operatore di digitare i dati richiesti. Le stringhe inserite con l'istruzione INPUT non devono essere racchiuse tra virgolette a meno che contengano delle virgole o degli spazi significativi.

Alla pressione del tasto RITORNO da parte dell'operatore, l'esecuzione del programma riprende.

Le istruzioni INPUT e LINE INPUT dispongono di istruzioni PRINT incorporate. Quando il programma incontra un'istruzione INPUT con una stringa racchiusa tra virgolette, riproduce automaticamente quella stringa (vedere l'istruzione INPUT).

La differenza principale tra le istruzioni INPUT e LINE INPUT consiste nel fatto che LINE INPUT accetta anche stringhe con all'interno caratteri speciali (del tipo virgole), senza richiedere l'uso di virgolette, mentre l'istruzione INPUT richiede che le virgolette vengano specificate.

Esempio 1

Per trovare il valore di un numero elevato al quadrato:

```
10 INPUT X
20 PRINT X "AL QUADRATO E'" X^2
30 END
RUN
?
```

Se in risposta al punto interrogativo si digita il numero 5:

```
5 AL QUADRATO E' 25
Ok
```

Esempio 2

Per trovare l'area di un cerchio conoscendone il raggio:

```
10 PI=3.14
20 INPUT "Misura del raggio?";R
30 A=PI*R^2
40 PRINT "L'area del cerchio è";A
50 PRINT
60 GOTO 20
RUN
Misura del raggio? 7.4
L'area del cerchio è 171.9464
```

Si noti che la riga 20 nell'esempio mostrato sopra utilizza l'istruzione PRINT incorporata in INPUT.

INPUT# (istruzione)

Funzione

Legge unità di dati in un file in ordine sequenziale e li assegna alle variabili del programma.

Sintassi

INPUT# *numerofile*, *elencovariabili*

Commenti

numerofile è il numero utilizzato quando il file è stato aperto per l'input.

elencovariabili contiene i nomi delle variabili da assegnare alle unità del file.

Le unità di dati del file appaiono esattamente come se in risposta all'istruzione INPUT fossero stati digitati dati tramite la tastiera

Il tipo di variabile deve corrispondere al tipo specificato dal nome della variabile.

Con l'enuciazione INPUT#, non appare alcun punto interrogativo, al contrario di quanto avviene con INPUT.

Valori numerici

Con i valori numerici, gli spazi e gli avanzamenti di riga (linefeed) che precedono il valore vengono ignorati. Il primo carattere incontrato (esclusi appunto spazi e avanzamenti di riga) viene interpretato come l'inizio del numero. Il numero viene chiuso da uno spazio, un ritorno di riga (carriage return), un avanzamento di riga (linefeed) o una virgola.

Valori alfanumerici

Se GW-BASIC cerca una determinata stringa in un file sequenziale, gli spazi e gli avanzamenti di riga che precedono la stringa vengono ignorati.

Se il primo carattere incontrato è una doppia virgoletta ("), la stringa sarà data da tutti i caratteri racchiusi tra di essa e la successiva doppia virgoletta.

L'inclusione di virgolette in una stringa racchiusa tra virgolette non è permessa. La seconda doppia virgoletta determina sempre la fine della stringa.

Se il primo carattere della stringa non è una virgoletta, la fine della stringa viene determinata da una virgola, da un ritorno di riga, da un avanzamento di riga o, in mancanza, dalla lettura di 255 caratteri (lunghezza massima della stringa).

Se si giunge al termine del file durante l'INPUT di un'unità di dati numerica o alfanumerica, quell'unità viene soppressa.

L'istruzione INPUT# può utilizzare anche file ad accesso casuale.

INPUT$ (funzione)

Funzione

Restituisce una stringa di *x* caratteri letti dalla tastiera o dal numero del file.

Sintassi

INPUT$(x[,[#]*numerofile*)]

Commenti

Se per input si utilizza la tastiera, i caratteri digitati non appaiono sullo schermo. Tutti i caratteri di controllo (eccetto CTRL-BREAK) vengono trasmessi. CTRL-BREAK arresta l'esecuzione di INPUT$.

Per la lettura dei file di comunicazioni è preferibile utilizzare l'istruzione INPUT$ anzichè INPUT e LINE INPUT, in quanto nelle comunicazioni tutti i caratteri ASCII potrebbero essere significativi. INPUT è l'istruzione meno adatta, in quanto quando incontra una virgola o un avanzamento di riga (linefeed) si arresta. LINE INPUT si arresta all'incontro di un ritorno di riga (carriage return).

INPUT$ permette l'assegnazione ad una stringa di tutti i caratteri letti. INPUT$ restituisce *x* caratteri dal numero del file o dalla tastiera.

Per ulteriori informazioni riguardanti le comunicazioni, consultare l'appendice F nel *Manuale dell'utente*.

Esempio 1

L'esempio che segue visualizza il contenuto di un file sequenziale in forma esadecimale:

```
10 OPEN"I" ,1,"DATI"
20 IF EOF (1) THEN 50
30 PRINT HEX$(ASC(INPUT$(1,#1)));
40 GOTO 20
50 PRINT
60 END
```

Esempio 2

```
.
.
.
100 PRINT "DIGITARE C PER CONTINUARE O S PER STOP"
110 X$=INPUT$(1)
120 IF X$="C" THEN 500
130 IF X$="S" THEN 700 ELSE 100
.
.
```

INSTR (funzione)

Funzione

Ricerca la prima stringa y$ in x$ e restituisce la posizione in cui la stringa è stata individuata.

Sintassi

INSTR([*n*,]**x$**,**y$**)

Commenti

L'offset facoltativo *n* imposta la posizione da cui iniziare la ricerca. Il valore predefinito di *n* è 1.

n deve rientrare nell'intervallo numerico da 1 a 255. Altrimenti, viene restituito l'errore `Chiamata di funzione illegale.`

INSTR restituisce 0 se:

- n>LEN(x$)
- x$ è nulla
- y$ non viene trovata

Se y$ è nulla, INSTR restituisce *n*.

x$ e y$ possono essere variabili alfanumeriche, espressioni alfanumeriche o caratteri alfanumerici.

Esempi

```
10 X$="ABCDEBXYZ"
20 Y$="B"
30 PRINT INSTR(X$,Y$);INSTR(4,X$,Y$)
RUN
        2        6
Ok
```

L'interprete cerca nella stringa "ABCDFBXYZ" e trova la prima B in seconda posizione. Dopodiché la ricerca riprende dalla posizione 4 (D) e viene individuata la lettera B in posizione 6. I tre caratteri rimanenti vengono ignorati, in quanto le condizioni impostate in riga 30 sono state tutte soddisfatte.

INT (funzione)

Funzione

Tronca un'espressione in un numero intero (cioè la arrotonda per difetto).

Sintassi

INT(x)

Commenti

I numeri negativi restituiscono il numero più basso che segue.

Anche le funzioni FIX e CINT restituiscono valori interi.

Esempi

```
PRINT INT(98.89)
       98
Ok
PRINT INT(-12.11)
      -13
Ok
```

IOCTL (istruzione)

Funzione

Permette a GW-BASIC di mandare una stringa di "dati di controllo" ad una driver periferica di carattere ogni qual volta essa viene aperta.

Sintassi

IOCTL[#]*numerofile*,*stringa*

Commenti

numerofile è il numero del file aperto per la driver periferica.

stringa è un'espressione alfanumerica valida che contiene dei caratteri che controllano il dispositivo.

I comandi IOCTL sono generalmente composti da 2 o 3 caratteri seguiti da un argomento alfanumerico facoltativo. Una stringa IOCTL può essere lunga fino a 255 caratteri, con comandi separati l'uno dall'altro tramite l'uso di punti e virgola.

Esempi

Se un utente avesse installato una driver per sostituire LPT1, e quella driver fosse in grado di impostare la lunghezza di pagina (il numero di righe da stampare su una pagina prima di eseguire l'avanzamento di pagina), le righe seguenti aprirebbero la nuova driver LPT1 ed imposterebbero la lunghezza di pagina a 66 righe:

```
OPEN "LPT1:" FOR OUTPUT AS #1
IOCTL #1,"PL66"
```

Le istruzioni seguenti aprono LPT1 con una lunghezza di pagina iniziale di 56 righe:

```
OPEN "\DEV\LPT1" FOR OUTPUT AS #1
IOCTL #1,"PL56"
```

IOCTL$ (funzione)

Funzione

Permette a GW-BASIC di leggere una stringa di "dati di controllo" da una driver periferica di carattere.

Sintassi

IOCTL$([#]*numerofile*)

Commenti

numerofile è il numero del file aperto per la periferica.

La funzione IOCTL$ viene generalmente usata per sapere se l'istruzione IOCTL è stata eseguita con successo o meno. Inoltre, IOCTL$ viene usata per ottenere informazioni riguardanti la periferica (ad esempio la sua larghezza dopo che un'istruzione IOCTL l'ha impostata).

Esempi

```
10 OPEN "\DEV\MIALPT" AS#1
20 IOCTYL#1,"GW"
30 'Salvare in WID
40 WID=VAL(IOCTL$(#1))
```

KEY (istruzione)

Funzione

Permette l'inserimento rapido di un massimo di 15 caratteri in un programma con la pressione di un solo tasto, mediante la ridefinizione dei tasti di funzione di GW-BASIC.

Sintassi

KEY *numerotasto,espressione*
KEY ,CHR$**(*esacodice*)+**CHR$(*codicetasto*)
KEY ON
KEY OFF
KEY LIST

Commenti

numerotasto è il numero del tasto da definire. L'opzione varia da 1 a 10 e da 15 a 20.

espressione è l'assegnazione del tasto. Si può usare qualsiasi stringa valida composta da 1 a 15 caratteri. Se una stringa supera il limite di lunghezza di 15 caratteri, i caratteri in eccedenza vengono ignorati. Le costanti devono essere racchiuse tra virgolette.

codicetasto (scan code) è la variabile che definisce il tasto da "catturare". L'appendice H nel *Manuale dell'utente* contiene un elenco dei codici di tasto.

esacodice è il codice esadecimale assegnato ai tasti sotto illustrati:

Tasto	*Codice esadecimale*
EXTENDED	&H80
CAPSLOCK	&H40
NUMLOCK (BREAK)	&H20
ALT	&H08
CTRL	&H04
SHIFT	&H01, &H02, &H03

I codici esadecimali possono essere combinati, come in &H03, che rappresenta entrambi i tasti SHIFT.

Inizialmente, i tasti di funzione vengono assegnati alle funzioni sotto elencate:

F1	LIST	F2	RUN<-
F3	LOAD"	F4	SAVE"
F5	CONT<-	F6	,"LPT1:"<-
F7	TRON<-	F8	TROFF<-
F9	KEY	F10	SCREEN 000<-

Avvertenza La freccia (<-) significa che dopo la pressione di ciascuno di questi tasti non è necessario premere il tasto RITORNO.

Tutti i 10 tasti possono essere ridefiniti. Alla pressione del tasto, i dati assegnati vengono inseriti nel programma.

`KEY No. tasto, "espressione alfanumerica"`

Assegna l'espressione alfanumerica al tasto specificato.

`KEY LIST`

Elenca i valori di tutti i dieci tasti sullo schermo. I valori vengono visualizzati interamente, cioè per tutti i 15 possibili caratteri.

`KEY ON`

Visualizza i primi sei caratteri dei valori del tasto nella 25ma riga dello schermo. Quando la larghezza di visualizzazione viene impostata a 40, vengono visualizzati cinque dei dieci tasti. Quando invece la larghezza viene impostata ad 80, vengono visualizzati tutti i tasti.

`KEY OFF`

Cancella la visualizzazione dei tasti alla 25ma riga, rendendola disponibile per l'uso con il programma. I tasti di funzione non vengono comunque disattivati.

Se il valore di *numerotasto* non rientra nell'intervallo da 1 a 10, o da 15 a 20, si verifica l'errore `Chiamata di funzione illegale`. La precedente assegnazione del tasto viene conservata.

L'assegnazione di una stringa nulla (di lunghezza 0) disattiva il tasto dalla funzione.

Alla ridefinizione del tasto di funzione, la funzione INKEY$ restituisce un carattere della stringa assegnata. Se il tasto di funzione viene disattivato, INKEY$ restituisce una stringa a due caratteri: il primo è uno zero binario e il secondo è il codice di tasto.

Esempi

```
10 KEY 1,"MENU"+CHR$(13)
```

Visualizza un menu selezionato dall'operatore ogni volta che il tasto 1 viene premuto.

```
1 KEY OFF
```

Disattiva la visualizzazione dei tasti.

```
10 DATA KEY1, KEY2, KEY3, KEY4, KEY5
20 FOR N=1 TO 5:READ SOFTKEYS$(n)
30 KEY N,SOFTKEYS$(I)
40 NEXT N
50 KEY ON
```

Visualizza i nuovi tasti di funzione sulla riga 25 dello schermo.

```
20 KEY 1,""
```

Disattiva il tasto di funzione 1.

KEY(n) (istruzione)

Funzione

Inizia e termina la "cattura" (o trapping) di tasti in un programma di GW-BASIC.

Sintassi

KEY(*n*)ON
KEY(*n*)OFF
KEY(*n*)STOP

Commenti

n è un numero da 1 a 20 che indica il tasto da "catturare". I tasti sono numerati nel seguente modo:

Numero del tasto	*Tasto*
1-10	I tasti di funzione da F1 a F10
11	SU
12	A DESTRA
13	A SINISTRA
14	GIU'
15-20	I tasti definiti nei formati seguenti (vedere l'istruzione KEY): KEY *n*,CHR$(*esacodice*)+CHR$(*codicetasto*)

L'esecuzione dell'istruzione KEY(*n*) ON è necessaria per attivare la "cattura" dei tasti di funzione o dei tasti di direzione. Quando l'istruzione KEY(*n*) ON viene attivata, GW-BASIC esegue il controllo di ciascuna nuova istruzione per verificare se è stato premuto il tasto specificato. Nel caso che quel tasto sia stato premuto, GW-BASIC esegue un GOSUB al numero di riga specificato nell'istruzione ON KEY(*n*). L'istruzione ON KEY(*n*) deve precedere un'istruzione KEY(*n*).

Dopo l'esecuzione di KEY(*n*) OFF, non avviene alcun trapping.

Se viene eseguita l'istruzione KEY(*n*) STOP, non avviene alcun trapping.

Tuttavia, se viene premuto il tasto specificato, la digitazione viene memorizzata in modo che all'eventuale successiva esecuzione dell'istruzione KEY(*n*) ON avvenga la "cattura" immediata di tale tasto.

Per ulteriori informazioni, si veda l'istruzione ON KEY (*n*).

KILL (comando)

Funzione

Cancella un file dal disco.

Sintassi

KILL *nomefile*

Commenti

nomefile può rappresentare un file programma o un file di dati sequenziale o ad accesso casuale.

Avvertenza Quando si usa il comando KILL, è necessario specificare l'estensione del nome del file. Ricordarsi che ai file salvati in GW-BASIC viene attribuita l'estensione predefinita .BAS.

Se si esegue il comando KILL per un file correntemente aperto, si verifica l'errore `File già aperto.`

Esempi

Il comando seguente cancella il file DATI1 creato da GW-BASIC, creando spazio disponibile per un altro file:

```
200 KILL "DATI1.BAS"
```

Il comando seguente cancella il file di GW-BASIC CLIENTI dalla subdirectory \UTENTE\GABRIELE:

```
KILL "\UTENTE\GABRIELE\CLIENTI.BAS"
```

LEFT$ (funzioni)

Funzione

Restituisce la stringa contenente gli *n* caratteri più a sinistra di x$.

Sintassi

LEFT$(x$,*n*)

Commenti

Il valore di *n* deve rientrare nell'intervallo che va da 0 a 255. Se *n* è maggiore di LEN(x$), viene restituita l'intera stringa (x$). Se il valore di *n* è zero, viene restituita una stringa nulla (di lunghezza 0). Vedere le funzioni MID$ e RIGHT$.

Esempi

```
10 A$="BASIC"
20 B$=LEFT$(A$,3)
30 PRINT B$
RUN
BAS
Ok
```

I primi tre caratteri dalla parte sinistra della parola "BASIC" vengono stampati sullo schermo.

LEN (funzione)

Funzione

Restituisce il numero dei caratteri contenuti in x$.

Sintassi

LEN(x$)

Commenti

Vengono considerati anche i caratteri non stampabili e gli spazi.

Esempi

```
10 X$="BERGAMO, ITALIA"
20 PRINT LEN(X$)
15
Ok
```

Notare che la virgola e lo spazio sono inclusi nel conteggio dei caratteri (che risultano essere 15).

LET (istruzione)

Funzione

Assegna il valore di un'espressione ad una variabile.

Sintassi

[LET] *variabile=espressione*

Commenti

L'inclusione della parola LET è facoltativo. Il simbolo di uguaglianza è sufficiente per assegnare un'espressione ad un nome di variabile.

L'istruzione LET viene usata raramente. E' stata inclusa in questa versione di GW-BASIC per assicurare la compatibilità con versioni precedenti di BASIC in cui l'uso di LET è necessario. Nell'uso di LET, ricordare che il tipo di variabile deve corrispondere al tipo di espressione. Altrimenti, si verifica l'errore `Tipi di carattere contrastanti.`

Esempio 1

L'esempio seguente garantisce la compatibilità con il vecchio sistema. Se tale compatibilità non è necessaria, usare il secondo esempio, che occupa meno memoria:

```
110 LET D=12
120 LET E=12^2
130 LET F=12^4
140 LET SUM=D+E+F
.
.
.
```

Esempio 2

```
110 D=12
120 E=12^2
130 F=12^2
140 SUM=D+E+F
.
.
.
```

LINE (istruzione)

Funzione

Traccia righe e quadrati sullo schermo.

Sintassi

LINE [(*x1,y1*)]-(*x2,y2*) [,[*attributo*][,**B**[**F**]][,*stile*]]

Commenti

x1,y1 e *x2,y2* specificano i limiti della linea da tracciare.

La modalità di risoluzione è determinata dall'istruzione SCREEN.

attributo specifica il colore o l'intensità del pixel visualizzato (si vedano COLOR e PALETTE).

B traccia un quadrato considerando *(x1,y1)* e *(x2,y2)* gli angoli opposti.

BF traccia un quadrato (analogamente a **B**) e ne riempie l'interno con dei punti.

Avvertenza Se *attributo* viene omesso, **B** o **BF** devono essere preceduti da due virgole.

LINE supporta l'argomento aggiuntivo *stile*. *stile* è una maschera intera a 16 bit la quale viene usata nell'inserimento dei pixel sullo schermo. Quest'operazione viene chiamata **line-styling**.

Ogni volta che LINE posiziona un punto sullo schermo, utilizza il bit di stile corrente. Se il bit di stile è 0, l'inserimento non avviene. Se invece ha il valore 1, l'inserimento viene effettuato normalmente. Dopo ciascun punto, viene selezionata una nuova posizione del bit di stile.

Dal momento che il bit di stile 0 non elimina il contenuto precedente, è consigliabile tracciare la riga di fondo prima della riga stilizzata, in modo da creare un fondo noto.

stile può essere usato per l'impostazione di righe e quadrati normali, ma non per l'impostazione di quadrati pieni.

Se il parametro **BF** e *stile* vengono usati in combinazione, appare il messaggio `Errore di sintassi.`

Quando nell'istruzione LINE vengono forniti valori oltre i limiti, le coordinate che oltrepassano quei limiti non sono visibili sullo schermo. Questa operazione viene chiamata **line-clipping**.

Nella sintassi qui illustrata, la forma di coordinata STEP *(x offset,y offset)* non è mostrata. Tuttavia, questa forma può essere usata ovunque vi sia una coordinata.

Se nell'istruzione LINE viene usata la forma relativa sulla seconda coordinata, tale forma è relativa alla prima coordinata.

Dopo un'istruzione LINE, l'ultimo punto utilizzato è *x2,y2*.

La sintassi di LINE più semplice è la seguente:

LINE - (*xz,yz*)

Essa traccia una riga nel colore di primo piano dall'ultimo punto utilizzato al punto (*xz,yz*)

Esempi

```
LINE (160,0)-(639,100)
```

Traccia una riga orizzontale che divide lo schermo in una metà superiore ed una inferiore, in SCREEN 2.

```
LINE (160,0)-(160,199)
```

Traccia una riga verticale che divide lo schermo in una metà destra ed una sinistra, in SCREEN 1; crea la divisione dello schermo in due parti, l'una 1/4 e l'altra 3/4, in SCREEN 2.

```
LINE (0,0)-(319,199)
```

Traccia una diagonale dall'angolo superiore sinistro dello schermo all'angolo inferiore destro in SCREEN 1, e dall'angolo superiore sinistro al centro del lato inferiore dello schermo in SCREEN 2.

```
LINE (10,10)-(20,20),2
```

Se SCREEN 1 è stato precedentemente specificato, traccia una riga nel colore 2 (si veda l'istruzione COLOR).

```
10 CLS
20 LINE -(RND*319,RND*199),RND*4
30 GOTO 20
```

Traccia righe ininterrottamente utilizzando attributi casuali.

```
10 FOR X=0 TO 319
20 LINE (X,0)-(X,199),X AND 1
30 NEXT
```

Traccia una struttura alternata: una riga sì e una riga no.

```
10 CLS
20 LINE -(RND*639,RND*199),RND*2,BF
30 GOTO 20
```

Traccia righe su tutto lo schermo.

```
LINE (0,0)-(100,175),,B
```

Traccia un quadrato nell'angolo superiore sinistro dello schermo.

```
LINE (0,0)-(100,175),,BF
```

Traccia lo stesso quadrato e lo riempie.

```
LINE (0,0)-(100,175),2,BF
```

Traccia lo stesso quadrato e lo riempie col colore magenta, in SCREEN 1.

```
LINE (0,0)-(100,350),,B
```

Se è stato specificato SCREEN 2, traccia lo stesso quadrato.

```
400 SCREEN 1
410 LINE(160,100)-(160,199),,,&HCCCC
```

Traccia una riga tratteggiata verticale al centro, in SCREEN 1.

```
220 SCREEN 2
230 LINE(300,100)-(400,50),,B,&HAAAA
```

Traccia un rettangolo usando una riga tratteggiata, in SCREEN 2.

```
LINE (0,0)-(160,100),3,,&HFF00
```

Traccia una riga tratteggiata dall'angolo superiore sinistro al centro dello schermo.

LINE INPUT (istruzione)

Funzione

Inserisce un'intera riga (fino a 255 caratteri) dalla tastiera in una variabile alfanumerica, ignorando i delimitatori.

Sintassi

LINE INPUT [;][*sollecito*;]*variabile*

Commenti

sollecito è una stringa visualizzata sullo schermo, la quale permette all'utente l'inserimento di dati durante l'esecuzione del programma.

Con questa istruzione, il punto interrogativo non appare, a meno che faccia parte della stringa di sollecito.

variabile è una stringa composta da tutto quanto viene digitato dalla fine del sollecito al ritorno di riga (carriage return). Gli spazi finali vengono ignorati.

Le istruzioni LINE INPUT ed INPUT sono simili; la sola differenza consiste nel fatto che LINE INPUT accetta l'inserimento di caratteri speciali (del tipo virgole) durante l'esecuzione del programma.

Se viene incontrata una sequenza di avanzamento/ritorno di riga (in quest'ordine), entrambi i caratteri vengono inseriti e visualizzati. Dopodiché l'inserimento della stringa può continuare.

Se LINE INPUT è immediatamente seguito da un punto e virgola, la pressione del tasto RITORNO non determina lo spostamento del cursore alla riga seguente.

L'interruzione dell'esecuzione di LINE INPUT viene effettuata digitando CTRL BREAK. GW-BASIC ritorna in questo caso al livello di comando e visualizza il prompt *Ok*.

Se si digita CONT, viene ripristinata l'esecuzione alla riga di LINE INPUT.

Esempio

```
100 LINE INPUT A$
```

Il programma fa una pausa alla riga 100, e tutti i caratteri della tastiera digitati da quel punto in poi vengono inseriti nella stringa A$, fino all'inserimento di RITORNO, CTRL-M, CTRL-C o CTRL-BREAK.

LINE INPUT# (istruzione)

Funzione

Legge un'intera riga (fino a 255 caratteri), senza delimitatori, da un file sequenziale in una variabile alfanumerica.

Sintassi

LINE INPUT# *numerofile*, *variabile*

Commenti

numerofile è il numero sotto il quale il file è stato aperto.

variabile è il nome della variabile alfanumerica alla quale viene assegnata la riga.

LINE INPUT# legge tutti i caratteri da un file sequenziale fino a che incontra un ritorno di riga (carriage return). Se incontra una sequenza avanzamento/ritorno di riga (in quest'ordine), la inserisce come ogni altro carattere.

LINE INPUT# è specialmente utile se ciascuna riga di un file di dati è stata spezzettata in campi, o se un programma di GW-BASIC salvato in modalità ASCII viene letto come insieme di dati da un altro programma.

Esempi

```
10 OPEN "O" ,1,"INFO"
20 LINE INPUT "INFORMAZIONI SUI CLIENTI?";C$
30 PRINT#1, C$
40 CLOSE 1
50 OPEN "I",1,"INFO"
60 LINE INPUT#1, C$
70 PRINT C$
80 CLOSE 1
RUN
INFORMAZIONI SUI CLIENTI?
```

Se l'operatore inserisce

```
SILVIA MAGGI        234,4       MILANO
```

Il programma continua con quanto segue:

```
SILVIA MAGGI        234,4       MILANO
Ok
```

LIST (comando)

Funzione

Elenca un programma, in tutto o in parte, sullo schermo, alla stampante di sistema o ad un file.

Sintassi

LIST [*numeroriga*][-*numeroriga*][,*nomefile*]
LIST [*numeroriga*-][,*nomefile*]

Commenti

numeroriga è un numero di riga valido nell'intervallo da 0 a 65529.

Se *nomefile* viene omesso, le righe specificate vengono elencate sullo schermo.

Il trattino viene usato per specificare un insieme di righe da elencare. Se il trattino viene omesso, viene elencato l'intero programma. *numeroriga*- elenca la riga specificata più tutte le righe che seguono. -*numeroriga* elenca le righe dall'inizio del programma alla riga specificata.

Il punto (.) può sostituire qualsiasi *numeroriga* ed indica la riga corrente.

La pressione di CTRL-BREAK permette l'interruzione di qualsiasi elencazione in corso.

Esempi

```
LIST
```

Elenca tutte le righe nel programma.

```
LIST -20
```

Elenca le righe da 1 a 20.

```
LIST 10-20
```

Elenca le righe da 10 a 20.

```
LIST 20-
```

Elenca le righe da 20 alla fine del programma.

LLIST (comando)

Funzione

Elenca tutto o parte di un programma correntemente in memoria alla stampante di sistema.

Sintassi

LLIST [*numeroriga*][-*numeroriga*]
LLIST [*numeroriga*-]

Commenti

Dopo l'esecuzione di LLIST, GW-BASIC ritorna sempre al livello di comando. Le opzioni per la definizione dell'intervallo di righe sono le stesse di LIST.

Esempi

Vedere gli esempi mostrati nell'istruzione LIST.

LOAD (comando)

Funzione

Carica in memoria un file da un disco.

Sintassi

LOAD *nomefile*[**,r**]

Commenti

nomefile è il nome sotto il quale il file è stato salvato. Se l'estensione viene omessa, viene usato .BAS.

Prima del caricamento del programma specificato, LOAD chiude tutti i file aperti e cancella tutte le variabili e righe del programma correntemente in memoria.

Se in combinazione con il comando LOAD viene usata l'opzione **r**, il programma viene eseguito immediatamente dopo il caricamento e tutti i file di dati aperti in quel momento vengono mantenuti aperti.

La combinazione del comando LOAD con l'opzione **r** permette il concatenamento di vari programmi (o segmenti dello stesso programma). L'uso del file di dati permette il passaggio di informazioni tra programmi.

Esempi

```
LOAD "PROGR1",R
```

Carica il file PROGR1.BAS e lo esegue, lasciando tutti i file aperti e le variabili del programma precedente intatti.

LOC (funzione)

Funzione

Restituisce la posizione corrente nel file.

Sintassi

LOC(*numerofile*)

Commenti

numerofile è il numero usato all'apertura del file.

Quando si trasmette o si riceve un file attraverso la porta di comunicazione, LOC restituisce il numero dei caratteri che attendono di essere letti, contenuti nel buffer di input. La dimensione predefinita per il buffer di input è di 256 caratteri, ma questo numero può essere modificato attraverso l'opzione **/c:** sulla riga di comando di GW-BASIC. Se il buffer contiene più di 255 caratteri, LOC restituisce 255. Dal momento che la capienza massima di una stringa è di 255 caratteri, questo limite elimina la necessità di verificare la dimensione della stringa prima di inserire i dati. Se il buffer contiene meno di 255 caratteri, LOC restituisce il numero effettivo.

Con file ad accesso casuale, LOC restituisce il numero del record appena scritto o letto con l'istruzione PUT o GET.

Con file sequenziali, LOC restituisce il numero dei blocchi da 128 byte letti o scritti sul file da quando è stato aperto. Quando un file sequenziale viene aperto per l'inserimento di dati, GW-BASIC legge inizialmente il primo settore del file. In questo caso, la funzione LOC restituisce il carattere *1* prima di permettere qualsiasi altro input.

Se il file viene aperto senza che sia avvenuto alcun input/output di disco, LOC restituisce uno zero.

Esempi

```
200 IF LOC(1)>50 THEN STOP
```

Il programma si arresta dopo la lettura o la scrittura di 51 record.

LOCATE (istruzione)

Funzione

Sposta il cursore alla posizione specificata nello schermo attivo. Parametri opzionali determinano l'intermittenza del cursore e definiscono i limiti di riga iniziali e finali per il cursore (raster lines). I limiti di riga rappresentano la distanza orizzontale o verticale tra due punti adiacenti indirizzabili sullo schermo.

Sintassi

LOCATE [*riga*][,[*col*][,[*cursore*][,[*inizio*] [,*fine*]]]]

Commenti

riga è il numero della riga ed è un'espressione numerica che rientra nell'intervallo da 1 a 25.

col è il numero della colonna dello schermo ed è un'espressione numerica che rientra nell'intervallo da 1 a 40, o da 1 a 80, a seconda della larghezza dello schermo.

cursore è un valore booleano che indica se il cursore è visibile (zero = OFF, non zero = ON).

inizio è la riga di limite iniziale del cursore ed è un'espressione numerica che rientra nell'intervallo da 0 a 31.

fine è la riga di limite finale del cursore ed è un'espressione numerica che rientra nell'intervallo da 0 a 31.

Quando il cursore viene spostato alla posizione specificata, le istruzioni PRINT seguenti iniziano a porre i caratteri in quella posizione. L'istruzione LOCATE può anche essere usata per rendere il cursore intermittente o per modificare la dimensione del cursore intermittente.

Qualsiasi valore inserito che non rientra nei limiti specificati determina l'errore `Chiamata di funzione illegale`. Vengono in questo caso conservati i valori precedenti.

Durante l'impostazione dei parametri per l'istruzione LOCATE, si potrebbe decidere di non cambiare una o più specificazioni già esistenti. Per l'omissione di un parametro, inserire una virgola al posto del parametro. Se l'omissione del parametro avviene alla fine dell'istruzione, la digitazione della virgola non è necessaria.

Se il parametro *inizio* viene fornito ed il parametro *fine* viene omesso, questo secondo parametro si suppone essere uguale a quello d'inizio.

Esempi

```
10 LOCATE 1,1
```

Sposta il cursore alla posizione iniziale nell'angolo superiore sinistro.

```
20 LOCATE ,,1
```

Rende visibile il cursore, la cui posizione resta comunque invariata. Notare che i primi due parametri non vengono usati; ciascuno di essi è stato sostituito da una virgola.

```
30 LOCATE ,,,7
```

La posizione e la visualizzazione del cursore rimangono intatte. Il cursore viene impostato in modo da apparire sotto al carattere che inizia e termina alla fine 7.

```
40 LOCATE 5,1,1,0,7
```

Sposta il cursore sulla riga 5, colonna 1, e conseguentemente ne esegue l'attivazione. Il cursore copre l'intera cella del carattere, che inizia dalla fine 0 e termina sulla fine 7.

LOCK (istruzione)

Funzione

Limita l'accesso da parte di un altro processo a tutto o a parte del file aperto. Questa protezione viene usata in un ambiente con una molteplicità di unità periferiche operanti, spesso chiamato **rete**.

Sintassi

LOCK [#]*n*[,[*numerorecord*] [**TO** *numerorecord*]]

Commenti

n è il numero originariamente assegnato al file nel programma.

numerorecord è il numero del record da proteggere. Nel caso vi sia un insieme di record da proteggere, *numerorecord* determina il record iniziale e finale dell'intervallo desiderato.

L'intervallo dei numeri di record accettabili va da 1 a 2^{32}-1. Il limite massimo per la dimensione di un record è di 32767 byte.

Nella specificazione di un intervallo, il limite inferiore deve precedere quello superiore.

Se il numero del record iniziale non viene specificato, viene usato il numero 1.

Se il numero del record finale non viene specificato, viene protetto soltanto il record specificato.

Gli esempi che seguono costituiscono istruzioni LOCK valide:

LOCK #*n*	protegge l'intero file *n*
LOCK #*n*,X	protegge soltanto il record X
LOCK #*n*, TO Y	protegge i record da 1 a Y
LOCK #*n*,X TO Y	protegge i record da X a Y

Con un file ad accesso casuale, si può proteggere l'intero file aperto, o un intervallo di record contenuti in quel file, rendendoli così inaccessibili a qualsiasi altro processo che abbia pure file aperti.

Con un file ad accesso sequenziale aperto per qualsiasi esecuzione di input o output, l'intero file viene protetto, senza badare ad eventuali record specificati. Ciò non costituisce un errore. La specificazione di un gruppo di record nell'istruzione LOCK riguardante un file sequenziale viene semplicemente ignorata.

L'istruzione LOCK dovrebbe essere eseguita in un file o in un intervallo di record contenuti in un file prima di tentare la lettura o la scrittura di quel file.

Prima di poter chiudere un file o un gruppo di record in un file precedentemente protetto bisogna togliere la protezione. Se l'istruzione UNLOCK non viene eseguita, si possono determinare future complicazioni nell'accesso a quel file in un ambiente di rete.

Si suppone normalmente che lo spazio di tempo durante il quale un file o un gruppo di record in un file rimangono protetti sia breve, per cui si consiglia di utilizzare la combinazione di istruzioni LOCK/UNLOCK per intervalli di tempo brevi.

Esempi

Il seguente esempio mostra come le istruzioni LOCK/UNLOCK dovrebbero essere usate:

```
LOCK #1, 1 TO 4
LOCK #1, 5 TO 8
UNLOCK #1, 1 TO 4
UNLOCK #1, 5 TO 8
```

L'esempio seguente non è accettabile:

```
LOCK #1, 1 TO 4
LOCK #1, 5 TO 8
UNLOCK #1, 1 TO 8
```

LOF (funzione)

Restituisce la lunghezza (numero di byte) assegnata al file.

Sintassi

LOF(*numerofile*)

Commenti

numerofile è il numero assegnato al file quando è stato aperto.

Con i file di comunicazioni, LOF restituisce la quantità di spazio libero nei buffer di input.

Esempi

Questo esempio ottiene l'ultimo record del file ad accesso casuale FILE.BIG, e suppone che il file sia stato creato con una lunghezza di record predefinita di 128 byte:

```
10 OPEN "R",1,"FILE.BIG"
20 GET #1,LOF(1)/128
.
.
.
```

LOG (funzione)

Funzione

Calcola il logaritmo naturale di *x*.

Sintassi

LOG(*x*)

Commenti

x deve essere un numero maggiore di zero.

LOG(*x*) viene calcolato in precisione singola, a meno che non venga usato il parametro **/d** all'esecuzione di GW-BASIC.

Esempi

```
PRINT LOG(2)
.6931471

PRINT LOG(1)
0
```

LPOS (funzione)

Funzione

Restituisce la posizione corrente della testina della stampante di sistema nel buffer della stampante.

Sintassi

LPOS(x)

Commenti

LPOS non fornisce necessariamente la posizione fisica della testina di stampa.

x è un argomento fittizio.

Se la stampante dispone di una capacità di stampa inferiore a 132 caratteri per riga, può creare avanzamenti di riga (linefeed) interni senza informare il buffer della stampante interna. Se ciò accade, il valore restituito da LPOS(x) potrebbe essere errato. Infatti, LPOS(x) conta semplicemente il numero di caratteri stampabili dopo l'ultimo avanzamento di riga.

Esempi

La riga seguente determina un ritorno di riga (carriage return) dopo la stampa del 60mo carattere sulla riga:

```
100 IF LPOS(X)>60 THEN LPRINT CHR$(13)
```

LPRINT e LPRINT USING (istruzioni)

Funzione

Trasmettono i dati alla stampante di sistema per la stampa.

Sintassi

LPRINT [*elencoespressioni*][;]
LPRINT USING *espressione*; *elencoespressioni*[;]

Commenti

elencoespressioni è un elenco composto da espressioni numeriche o alfanumeriche separate da punti e virgola.

espressione è una stringa letterale o variabile composta da caratteri speciali di formattazione. I caratteri di formattazione determinano il campo ed il formato di stringhe o di numeri stampati.

Queste istruzioni sono simili a PRINT e PRINT USING, con la differenza che LPRINT e LPRINT USING mandano l'output alla stampante di sistema. Per ulteriori informazioni riguardanti i campi numerici e alfanumerici e le variabili usate, consultare le istruzioni PRINT e PRINT USING.

Le istruzioni LPRINT e LPRINT USING assumono che si disponga di una stampante di sistema a 80 caratteri per riga.

Per la reimpostazione dei caratteri che si desiderano stampare orizzontalmente sulla pagina (nell'ipotesi che la stampante permetta l'inserimento di più di 80 caratteri per riga), si veda l'istruzione WIDTH.

LSET e RSET (istruzioni)

Funzione

Spostano i dati dalla memoria al buffer del file ad accesso casuale allineandoli a destra o a sinistra per prepararli ad un'istruzione PUT.

Sintassi

LSET *variabile =espressione*
RSET *variabile =espressione*

Commenti

Se l'*espressione* (alfanumerica) richiede meno byte di quanti sono stati definiti per la *variabile* (alfanumerica), LSET allinea la stringa a sinistra del campo, mentre RSET allinea la stringa a destra (per riempire le posizioni che avanzano vengono aggiunti spazi).

Se la stringa è troppo lunga per il campo, vengono troncati dei caratteri iniziando dalla destra.

Per convertire i valori numerici in stringhe prima dell'uso dell'istruzione LSET o RSET, vedere le funzioni MKI$, MKS$, e MKD$.

LSET e RSET possono anche essere usate con variabili alfanumeriche prive della definizione del campo, per essere allineate una stringa a destra o a sinistra di una determinata zona.

Esempi

```
110 A$=SPAZIO$(20)
120 RSET A$=N$
```

Queste due istruzioni impostano l'allineamento a destra della stringa N$ in un campo di 20 caratteri. Questo può essere utile per la formattazione dell'output di stampa.

MERGE (comando)

Funzione

Fonde le righe di un programma ASCII con quelle di un programma già in memoria.

Sintassi

MERGE *nomefile*

Commenti

nomefile è un'espressione alfanumerica valida contenente il nome del file. Se l'estensione non viene specificata, GW-BASIC utilizza l'estensione .BAS.

Il file specificato viene cercato nel dischetto. Se non viene trovato, le righe del programma sul dischetto vengono fuse con quelle in memoria. Dopo il comando MERGE, il programma risultante viene inserito in memoria, per poi permettere a GW-BASIC di ritornare in modalità diretta.

Se il programma da unire a quello in memoria non è stato salvato in codice ASCII (opzione **a** col comando SAVE), si verifica l'errore `Modalità del file errata`. Il programma in memoria rimane intatto.

Se uno dei numeri di riga nel file corrisponde ad uno dei numeri di riga nel programma in memoria, le righe del file sostituiscono le corrispondenti righe nel programma in memoria.

Esempi

```
MERGE "SUBRTN"
```

Fonde il file SUBRTN.BAS con il programma residente in memoria, ammesso che SUBRTN sia stato precedentemente salvato con l'opzione **a**. Se alcune delle righe del programma in memoria corrispondono a righe del file SUBRTN.BAS, le righe del programma originale vengono sostituite da quelle corrispondenti nel file SUBRTN.BAS.

MID$ (funzione)

Funzione

Restituisce una stringa di *m* caratteri da x$, iniziando dalla posizione *n*.

Sintassi

MID$(x$,*n*[,*m*])

Commenti

n deve rientrare nell'intervallo da 1 a 255.

m deve rientrare nell'intervallo da 0 a 255.

Se viene omessa *m*, o se vi sono meno di *m* caratteri alla destra di *n*, tutti i caratteri posti a destra di *n* (compreso) vengono restituiti.

Se *n* > LEN(x$), la funzione MID$ restituisce una stringa nulla.

Se *m* è uguale a 0, la funzione MID$ restituisce una stringa nulla.

Se *n* o *m* non rientrano nei limiti numerici sopra specificati, viene restituito l'errore `Chiamata di funzione illegale.`

Per ulteriori informazioni ed esempi, vedere le funzioni LEFT$ e RIGHT$.

Esempi

```
10 A$="COLORE"
20 B$="ROSSO VERDE GIALLO"
30 PRINT A$;MID$(B$,6,6)
RUN
COLORE VERDE
Ok
```

La riga 30 concatena la stringa A$ con un'altra stringa della lunghezza di 6 caratteri, iniziando dalla posizione 6, nella stringa B$.

MID$ (istruzione)

Funzione

Sostituisce una porzione di una stringa con un'altra stringa.

Sintassi

MID$(*espressione1*,*n*[,*m*])=*espressione2*

Commenti

Sia *n* che *m* sono valori interi.

espressione1 ed *espressione2* sono espressioni alfanumeriche (stringhe).

I caratteri contenuti in *espressione1*, iniziando dalla posizione *n*, vengono sostituiti dai caratteri in *espressione2*.

La *m* (facoltativa) indica il numero dei caratteri di *espressione2* che verranno usati. Se *m* viene omessa, vengono usati tutti i caratteri contenuti in *espressione2* dopo la posizione *n*.

Indipendentemente dall'inclusione o meno di *m*, la sostituzione dei caratteri non va mai oltre la lunghezza originale di *espressione1*.

Esempi

```
10 A$="MILANO, FRA"
20 MID$(A$,9)="ITA"
30 PRINT A$
RUN
MILANO, ITA
Ok
```

Con riga 20 si sovrappone "ITA" a "FRA" nella stringa A$.

MKDIR (comando)

Funzione

Crea una subdirectory.

Sintassi

MKDIR *nomepercorso*

Commenti

nomepercorso è un'espressione alfanumerica che non eccede i 63 caratteri ed identifica la subdirectory creata.

Esempi

```
MKDIR "C:\FATTURE\CLAUDIO"
```

Crea una subdirectory chiamata CLAUDIO all'interno della directory FATTURE.

MKI$, MKS$, MKD$ (funzioni)

Funzione

Convertono valori numerici in valori alfanumerici.

Sintassi

MKI$(*espressione intera*)
MKS$(*espressione a precisione singola*)
MKD$(*espressione a precisione doppia*)

Commenti

MKI$ converte un valore intero in una stringa a 2 byte.

MKS$ converte un numero a precisione singola in una stringa a 4 byte.

MKD$ converte un numero a precisione doppia in una stringa a 8 byte.

Qualsiasi valore numerico inserito in un buffer di file ad accesso casuale con l'istruzione LSET o RSET deve essere convertito in stringa (vedere CVI, CVS, CVD).

Queste funzioni differiscono da STR$ in quanto non eseguono l'effettivo cambio dei byte, ma ne cambiano soltanto le interpretazioni.

Esempi

```
90 AMT=(K+T)
100 FIELD #1,8 AS d$,20 AS N$
110 LSET D$=MKS$(AMT)
120 LSET N$=A$
130 PUT #1
.
.
.
```

NAME (comando)

Funzione

Cambia il nome di un file sul disco.

Sintassi

NAME *vecchio nomefile* **AS** *nuovo nomefile*

Commenti

vecchio nomefile deve già esistere e *nuovo nomefile* non deve esistere ancora; in caso contrario, si verifica un errore.

Dopo il comando NAME, il file si trova sullo stesso dischetto, nella stessa posizione, ma con un nome nuovo.

Esempi

```
NAME "CLIENTI" AS "VENDITE"
Ok
```

Il file in precedenza chiamato CLIENTI viene qui cambiato in VENDITE. I contenuti del file e la posizione fisica sul dischetto rimangono invariate.

NEW (comando)

Funzione

Cancella il programma correntemente in memoria e tutte le variabili utilizzate.

Sintassi

NEW

Commenti

NEW viene inserito al livello di comando per liberare la memoria prima dell'inserimento di un nuovo programma. Dopo questa operazione, GW-BASIC ritorna al livello di comando.

Esempi

```
NEW
OK
```

oppure

```
980 PRINT "Uscita (S/N)"
990 ANS$=INTST$: IF ANS$=""THEN 990
1000 IF ANS$="S" THEN NEW
1010 IF ANS$="N" THEN 980
1020 GOTO 990
```

OCT$ (funzione)

Funzione

Converte un valore decimale in un valore ottale.

Sintassi

OCT$(*x*)

Commenti

Prima che OCT$(x) sia calcolato, *x* viene arrotondato ad un valore intero.

Questa istruzione converte un valore decimale che rientra nell'intervallo da -32767 a +65535 in un'espressione alfanumerica ottale.

I numeri ottali sono numeri con base 8 anziché 10 (numeri decimali).

Per la conversione in forma esadecimale, si veda la funzione HEX$.

Esempi

```
10 PRINT OCT$(18)
RUN
22
Ok
```

Il valore decimale 18 equivale al valore ottale 22.

ON COM(n), ON KEY(n), ON PEN, ON PLAY(n), ON STRIG(n), e ON TIMER(n) (istruzioni)

Funzione

Crea un sistema di "cattura" (trapping) di particolari eventi (del tipo comunicazioni, pressione di tasti di funzione o di direzione, l'uso della penna luminosa, o l'uso di un joystick).

Sintassi

ON *indicatore* **GOSUB** *numeroriga*

Commenti

La sintassi sopra mostrata imposta il sistema di cattura (trapping) al numero di riga specificato per l'evento descritto. Se come *numeroriga* viene specificato 0, il sistema viene disattivato.

Una volta impostati i numeri di riga, il trapping può essere controllato dalle righe di sintassi seguenti:

indicatore **ON** — Quando un evento è ON, e il *numeroriga* non è uguale a zero, ogni volta che BASIC avvia un'istruzione nuova, esegue un controllo per constatare se l'evento specificato si è verificato. In caso affermativo, BASIC esegue un GOSUB alla riga specificata nell'istruzione.

indicatore **OFF** — Quando un evento è OFF, non avviene alcun trapping: anche se l'evento si verifica, non viene registrato dal computer.

indicatore **STOP** — Quando un evento è bloccato (STOP), non avviene alcun trapping; tuttavia, se quell'evento si verifica, ne avviene la memorizzazione in modo che all'attivazione (ON) l'evento venga immediatamente "catturato".

Conseguentemente al trapping di un evento, avviene uno stop automatico in

quell'evento in modo da evitare il verificarsi in modo ricorsivo del trapping.

A meno che non sia già stato eseguito un OFF, al ritorno da una routine di trapping viene automaticamente eseguito un ON.

Un errore di trapping determina l'automatica disattivazione del sistema di trapping.

Il trapping non viene effettuato se BASIC non sta eseguendo alcun programma.

I valori seguenti sono *indicatori* validi:

COM(*n*) — *n* è il numero del canale COM (1 o 2).

KEY(*n*) — *n* è un numero di tasto di funzione, da 1 a 20. I numeri da 1 a 10 sono i tasti di funzione da F1 a F10. I numeri da 11 a 14 rappresentano i tasti di direzione nell'ordine seguente:

11 = SU	13 = A DESTRA
12 = A SINISTRA	14 = GIU'

I tasti da 15 a 20 sono definiti dall'utente.

PEN — Dal momento che esiste soltanto una penna, non è necessario alcun numero.

PLAY(*n*) — *n* è un valore intero nell'intervallo da 1 a 32. I valori che non rientrano in questi limiti causano errori di `Chiamata di funzione illegale`.

STRIG(*n*) — *n* è 0, 2, 4 o 6. (0=trigger A1; 4=trigger A2; 2=trigger B1; 6=trigger B2).

TIMER(*n*) — *n* è un valore numerico tra 1 a 86400. Un valore che non rientra in questi limiti determina l'errore `Chiamata di funzione illegale`.

RETURN *numeroriga* — Questa forma di RETURN viene principalmente usata per il trapping di eventi. Pur eliminando il GOSUB creato, la routine di trapping potrebbe aver bisogno di ritornare ad un numero di riga specifico nel programma.

Il RETURN non locale deve essere usato attentamente. Qualsiasi GOSUB, WHILE o FOR attivo al momento del trapping rimane attivo.

Se il trapping esce dalla subroutine, qualsiasi tentativo di continuare i cicli al di fuori della subroutine determina l'errore `NEXT senza FOR`.

Avvertenze speciali riguardanti ciascun tipo di trapping

COM

Generalmente, prima di ritornare, la routine di trapping COM esegue la lettura di un intero messaggio dalla porta COM.

E' consigliabile non usare il trapping COM per i messaggi di un solo carattere, in quanto, ad un tasso di trasmissione elevato, il tempo assorbito dal trapping e dalla lettura di ciascun carattere potrebbe determinare un overflow del buffer di memoria.

KEY

I tasti "catturabili" da 15 a 20 sono definiti con l'istruzione seguente:

KEY(*n*),**CHR$**[*esacodice*]+**CHR$**[*codicetasto*]

n è un'espressione intera tra 15 e 20 che definisce il tasto da "catturare".

esacodice è il codice esadecimale, cioè la maschera del tasto bloccato: (CAPSLOCK, NUMLOCK, ALT, CTRL, SHIFT SINISTRO, SHIFT DESTRO)

codicetasto è il numero che identifica uno degli 83 tasti da "catturare". Per un elenco dei codici di tasto, consultare l'appendice H nel *Manuale dell' utente*.

Per "catturare" un tasto che viene premuto insieme ai tasti CTRL, ALT o SHIFT, bisogna impostare il bit appropriato in *esacodice*. I valori dei codici esadecimali sono i seguenti:

Maschera	*Codice esadecimale*	*Indica che*
EXTENDED	&H80	Il tasto viene esteso
CAPSLOCK	&H40	CAPSLOCK è attivo
NUMLOCK	&H20	NUMLOCK è attivo
ALT	&H08	Il tasto ALT è premuto
CTRL	&H04	Il tasto CTRL è premuto
SHIFT SINISTRO	&H02	Il tasto SHIFT sinistro è premuto
SHIFT DESTRO	&H01	Il tasto SHIFT destro è premuto

Per il trapping dei tasti "shiftati", si può usare il valore &H01, &H02, o &H03. Usando &H03 vengono considerati entrambi i tasti SHIFT.

Per ulteriori informazioni, si veda l'istruzione KEY(n).

Quando GW-BASIC si trova in modalità diretta, nessun tipo di trapping può essere attivato. In fase di input, i tasti di funzione riprendono il loro significato predefinito.

Un tasto che determina un trapping non può essere esaminato dall'istruzione INPUT o INKEY$; quindi, se si desidera un trattamento diverso, la routine di trapping per ciascun tasto deve essere diversa.

Se CTRL-PRTSC viene "catturato", l'eco alla stampante di sistema viene eseguito prima. La definizione di CTRL-PRTSC come trapping di tasto non impedisce la trasmissione dei caratteri alla stampante che segue la pressione di CTRL-PRTSC.

I tasti di funzione da 1 a 14 sono predefiniti. Quindi, l'impostazione dei codici di tasto 59-68, 72, 75, 77 o 80 non ha effetto.

PLAY(*n*)

Un trapping PLAY viene inserito soltanto in presenza di musica di fondo (PLAY"MB..). I trapping PLAY non vengono inseriti durante l'esecuzione di musica di primo piano (valore predefinito, o PLAY"MF..).

E' consigliabile scegliere valori non troppo alti per *n*. L'istruzione ON PLAY(32).. determina il trapping di eventi così frequenti da lasciare poco tempo per l'esecuzione del resto del programma.

L'istruzione ON PLAY(n) determina un trapping di evento quando la coda della musica di fondo va da *n* a *n* -1 note.

STRIG

L'uso di STRIG(*n*) ON attiva la routine d'interruzione che controlla lo stato del grilletto (trigger). I tasti che determinano il trapping non hanno alcuna influenza sulle funzioni STRIG(0), STRIG(2), STRIG(4) o STRIG(6).

TIMER(n)

L'istruzione ON TIMER(n) viene usata con le applicazioni che necessitano di un timer interno. Il trapping avviene a *n* secondi dall'esecuzione dell'istruzione TIMER ON.

Esempio 1

Questo è un semplice programma di simulazione di terminale.

```
10 REM ESEMPIO DI "ON COM(n)"
20 OPEN "COM1:9600,0,7" AS #1
30 ON COM(1) GOSUB 80
40 COM(1) ON
50 REM TRASMETTERE I CARATTERI DALLA TASTIERA
60 A$=INKEY$:IF A$=""THEN 50
70 PRINT #1,A$;:GOTO 50
80 REM VISUALIZZA I CARATTERI
90 ALL=LOC(1):IF ALL<1 THEN RETURN
100 B$=INPUT$(ALL,#1):PRINT B$;:RETURN
```

Esempio 2

Quanto segue impedisce un CTRL-BREAK od una reimpostazione del sistema durante l'esecuzione di un programma.

```
10 KEY 15,CHR$(&H04)+CHR$($H46) REM Trapping ^BREAK
20 KEY 16,CHR$(&HC)+CHR$(&H53) REM Trapping
   inizializzazione
30 ON KEY(15) GOSUB 1000
40 ON KEY(16) GOSUB 2000
50 KEY(15) ON
60 KEY(16) ON
.
.
.
1000 PRINT "Sono spiacente, ma non posso permetterlo"
1010 RETURN
2000 ATTEMPTS=ATTEMPTS+1
2010 ON ATTEMPTS GOTO 2100,2200,2300,2400,2500
2100 PRINT "Maria aveva un piccolo agnello":RETURN
2200 PRINT "Il suo pelo era bianco come la
     neve":RETURN
2300 PRINT "E seguiva Maria":RETURN
2400 PRINT "ovunque lei andasse":RETURN
2500 KEY(16) OFF REM Un'altra volta...
2510 RETURN REM e BASIC muore...
```

Esempio 3

Quanto segue visualizza l'orario alla riga 1 ogni minuto.

```
10 ON TIMER(60) GOSUB 10000
20 TIMER ON
.
.
.
10000 RIGVEC=CSRRIG REM Salva la riga corrente
10010 COLVEC=POS(0) REM Salva la colonna corrente
10020 LOCATE 1,1:PRINT TIME$
10030 LOCATE RIGVEC,COLVEC REM Ripristina la riga e la
colonna
10040 RETURN
```

ON ERROR GOTO (istruzione)

Funzione

Permette il trapping di errori e specifica la prima riga della subroutine per il relativo trattamento.

Sintassi

ON ERROR GOTO *numeroriga*

Commenti

Una volta impostato il trapping di errori, tutti gli errori trovati da GW-BASIC, compresi gli errori in modalità diretta (ad esempio, errori di sintassi), determinano il ricollegamento di GW-BASIC alla riga d'inizio della subroutine per il trattamento degli errori stessi.

GW-BASIC si ricollega alla riga specificata dall'istruzione ON ERROR finché incontra un'istruzione RESUME.

Se *numeroriga* non esiste, si verifica l'errore `Riga non definita.`

Per disattivare il trapping, eseguire l'istruzione seguente:

```
ON ERROR GOTO 0
```

Gli errori che seguono quest'istruzione determinano un messaggio d'errore e l'arresto dell'esecuzione.

Un'istruzione ON ERROR GOTO 0 inserita in una subroutine per il trattamento di errori determina l'arresto di GW-BASIC seguito dal messaggio relativo all'errore che ha determinato il ricollegamento alla subroutine. E' consigliabile fare in modo che tutte le subroutine per il trattamento di errori eseguano un ON ERROR GOTO 0, quando si incontra un errore per il quale non esiste possibilità di recupero.

Se si verifica un errore durante l'esecuzione di una subroutine per il trattamento di errori, viene stampato il messaggio d'errore e l'esecuzione viene terminata. Il trapping di errori non opera all'interno di una routine per il trattamento di errori.

Esempi

```
10 ON ERROR GOTO 1000
.
.
.
1000 A=ERR:B=ERL
1010 PRINT A,B
1020 RESUME NEXT
```

Per effetto della riga 1010 viene riprodotto il tipo e la posizione dell'errore sullo schermo (vedere le variabili ERR e ERL).

In riga 1020 si indica di continuare l'esecuzione del programma dalla riga che segue l'errore.

ON ... GOSUB e ON ... GOTO (istruzioni)

Funzione

Ricollegano il programma ad uno dei diversi numeri di riga specificati, in relazione al valore restituito in seguito al calcolo di un'espressione.

Sintassi

ON *espressione* **GOTO** *numeririga*
ON *espressione* **GOSUB** *numeririga*

Commenti

Nell'istruzione ON ... GOTO, il valore di *espressione* determina il numero di riga al quale ricollegarsi. Ad esempio, se il valore è 3, la destinazione è il terzo numero di riga nell'elenco. Se il valore non è intero, la porzione frazionale viene arrotondata.

Nell'istruzione ON ... GOSUB, ogni singolo numero di riga nell'elenco deve essere la prima riga di una subroutine.

Se il valore di *espressione* è zero o maggiore del numero di elementi nell'elenco (non oltre 255), GW-BASIC continua con l'istruzione eseguibile che segue.

Se il valore di *espressione* è negativo, o maggiore di 255, si verifica l'errore `Chiamata di funzione illegale`.

Esempi

```
100 IF R<1 or R>4 THEN PRINT "ERRORE":END
```

Se il valore intero di R è minore di 1 o maggiore di 4, l'esecuzione del programma viene terminata.

```
200 ON R GOTO 150,300,320,390
```

Se R=1, il programma si ricollega alla riga 150.

Se R=2, il programma si ricollega alla riga 300 continuando da quel punto. Se R=3, il ricollegamento avviene alla riga 320, e così via.

OPEN (enunciazione)

Funzione

Stabilisce un collegamento input/output (I/O) con un file o un'unità periferica.

Sintassi

OPEN *modalità*,[#]*numerofile*,*nomefile*[*lungrec*]

OPEN *nomefile* [**FOR** *modalità*][**ACCESS** *accesso*]
[*bloccaggio*]**AS**[#]*numerofile*[**LEN**=*lungrec*]

Commenti

nomefile è il nome del file.

modalità (nella prima sintassi) è un'espressione alfanumerica con uno dei seguenti caratteri:

Espressione	*Specifica*
O	Modalità output sequenziale
I	Modalità input sequenziale
R	Modalità input/output casuale
A	Fine del file (aggiunta)

modalità (nella seconda sintassi) determina il posizionamento iniziale all'interno di un file, e l'azione da eseguire nel caso il file non esistesse. Se la clausola di modalità FOR viene omessa, la posizione iniziale predefinita è l'inizio del file. Se il file non viene trovato, viene creato. Questa è la modalità I/O casuale. Vale a dire, i record possono essere letti o scritti in qualsiasi posizione del file. Le modalità valide e le azioni eseguite sono le seguenti:

INPUT	Si pone all'inizio del file. Si verifica l'errore `File non trovato` se il file non esiste.
OUTPUT	Record posti all'inizio del file. Se il file non esiste, viene creato.

APPEND	Record posti alla fine del file. Se il file non esiste, viene creato.
RANDOM	Specifica la modalità di input o di output casuale.

modalità deve essere una costante alfanumerica non racchiusa tra virgolette. *accesso* può essere dato da una delle seguenti stringhe:

READ

WRITE

READ WRITE

numerofile è un numero tra 1 ed il numero massimo di file permessi. Questo numero associa un buffer di I/O con un file di disco o un'unità periferica. L'associazione rimane impostata fino all'esecuzione di CLOSE o CLOSE *numerofile*.

lungrec è un'espressione intera compresa nell'intervallo da 1 a 32767 che imposta la lunghezza del record da usare per i file ad accesso casuale. Se si omette *lungrec*, la lunghezza del record è impostata a 128 byte.

Quando *lungrec* viene usato per i file sequenziali, il valore predefinito è 128 byte. Inoltre, *lungrec* non può in questo caso eccedere il valore specificato dal parametro /s.

Prima di poter eseguire qualsiasi operazione di I/O, il file deve essere aperto. L'istruzione OPEN assegna un buffer di I/O al file e determina la modalità di accesso da usarsi con il buffer.

E' possibile l'apertura simultanea di più file con numeri diversi, per operazioni di input o di accesso casuale. Ad esempio, le enunciazioni seguenti sono permesse:

```
OPEN "B:TEMP" FOR INPUT AS #1
OPEN "B:TEMP" FOR INPUT AS #2
```

Comunque, un determinato file può essere aperto una sola volta per operazioni di output o di aggiunta. Ad esempio, le istruzioni seguenti non sono accettabili:

```
OPEN "TEMP" FOR OUTPUT AS #1
OPEN "TEMP" FOR OUTPUT AS #2
```

Avvertenza Prima di estrarre i dischetti dalle unità, assicurarsi di aver chiuso tutti i file (si vedano CLOSE e RESET).

L'unità periferica può essere una delle seguenti:

A:,B:,C:...	Unità disco
KYBD:	Tastiera (solo per input)
SCRN:	Schermo (solo per output)
LPT1:	Stampante di sistema 1
LPT2:	Stampante di sistema 2
LPT3:	Stampante di sistema 3
COM1:	Comunicazioni RS-232 numero 1
COM2:	Comunicazioni RS-232 numero 2

Per ciascuna unità, sono permesse le modalità OPEN seguenti:

KYBD:	Solo input
SCRN:	Solo output
LPT1:	Solo output
LPT2:	Solo output
LPT3:	Solo output
COM1:	Solo input, output o casuale
COM2:	Solo input, output o casuale

I file di disco permettono tutte le modalità.

Quando un file di disco viene aperto per APPEND, la posizione viene inizialmente impostata alla fine del file, ed il numero di record viene impostato all'ultimo record del file (LOF(x)/128). Quindi, il file viene allargato da PRINT, WRITE o PUT. Il programma potrebbe posizionarsi in un punto diverso del file con l'istruzione GET. Se ciò avviene, la modalità viene cambiata in casuale e la posizione si sposta al record indicato.

Una volta spostata la posizione alla fine del file, possono essere aggiunti altri record con l'istruzione GET #x, LOF(x)/*lungrec*. Questo posiziona il puntatore alla fine del file in attesa dell'aggiunta.

Qualsiasi valore fornito che non rientra nei limiti impostati determina errori di `Chiamata di funzione illegale`. I file non vengono in questo caso aperti.

Se il file viene aperto per l'INPUT, qualsiasi tentativo di scrittura al file determina l'errore `Modalità del file errata`.

Se il file viene aperto per l'OUTPUT, qualsiasi tentativo di lettura determina un errore di `Modalità del file errata`.

Se un file è già aperto in qualsiasi modalità, l'apertura di quel file in OUTPUT o per l'aggiunta (APPEND) di dati non viene permessa.

Essendo possibile riferirsi allo stesso file contenuto in una subdirectory tramite percorsi diversi, a GW-BASIC riesce quasi impossibile riconoscere che si tratta dello stesso file, basandosi unicamente sul percorso. Ecco perchè, nonostante i percorsi siano differenti, GW-BASIC non permette l'apertura di un file sullo stesso disco per l'OUTPUT o per l'aggiunta (APPEND). Ad esempio se MARIA è la directory funzionante, tutte le istruzioni che seguono si riferiscono allo stesso file:

```
OPEN "RAPPORTO"
OPEN "\FATTURE\MARIA\RAPPORTO"
OPEN "..\MARIA\RAPPORTO"
OPEN "..\..\MARIA\RAPPORTO"
```

In qualsiasi momento, è possibile avere il nome di un file su un particolare dischetto aperto sotto più di un numero di file. Ciascun numero di file possiede un buffer diverso, per cui diversi record di un file possono essere mantenuti in memoria per un rapido accesso. Questo permette l'uso di modalità diverse per funzioni diverse; oppure, per chiarezza del programma, l'uso di numeri di file diversi per modalità di accesso diverse.

Se viene usata l'opzione LEN=*lungrec*, *lungrec* non può eccedere il valore impostato dall'opzione del parametro /s:*lungrec* nella riga di comando.

In rete, l'uso dell'istruzione OPEN viene basato su due diversi tipi di circostanze:

- Unità periferiche possono essere condivise soltanto per funzioni specifiche, in modo che le istruzioni OPEN possano essere limitate a modalità specifiche tra quelle che potrebbero essere richieste; ad esempio: INPUT, OUTPUT, APPEND e quella predefinita (casuale).
- I file possono essere limitati da un'istruzione OPEN che permette, attraverso un altro processo, il blocco di un file aperto con successo. Il blocco imposta dei limiti a quel file in modo da escluderlo dal processo quando l'istruzione OPEN è attiva.

bloccaggio può essere uno dei seguenti:

SHARED	La modalità "nessun divieto". Nessuna restrizione viene imposta sulla possibilità di accesso per lettura o scrittura del file in un altro processo. L'unica restrizione consiste nel fatto che la modalità predefinita non è ammessa da nessuna modalità comprendente SHARED.
LOCK READ	La modalità "lettura vietata". Una volta effettuata l'apertura di un file con accesso LOCK READ, nessun altro processo può avere accesso al file per la lettura. L'eventuale tentativo di aprire un file in questa modalità non potrà avere successo, se quel file è aperto in modalità predefinita o con accesso alla lettura.
LOCK WRITE	La modalità "scrittura vietata". Un file aperto con accesso LOCK WRITE non può essere aperto per l'accesso alla scrittura da un altro processo. L'eventuale tentativo di aprire un file in questa modalità non potrà avere successo, se quel file è aperto in modalità predefinita o con accesso alla scrittura.
LOCK READ WRITE	La modalità "divieto totale" o "esclusiva". Se un file viene aperto con questo accesso, il processo ha accesso esclusivo a quel file. Un file correntemente aperto in questa modalità non può essere riaperto in altra modalità da nessun processo.
predefinita	La modalità "compatibilità", con la quale è garantita la compatibilità con altri BASIC. Nessun tipo di accesso viene specificato. Il file può essere aperto da un processo più volte, sempre che il file non sia correntemente aperto da un altro processo. Quando il file è aperto in questa modalità, gli altri processi non possono avere accesso. La modalità è quindi funzionalmente esclusiva.

Il tentativo di aprire un file a cui un altro processo ha già avuto accesso, determina l'errore `Permesso negato`. Un esempio di situazione che genera questo errore è il tentativo da parte di un processo di aprire un file in modalità SHARED quando il file è già aperto in LOCK/READ/WRITE da un altro processo.

Se un'istruzione OPEN non ha successo a causa dell'incompatibilità della modalità con l'accesso condiviso impostato su un'unità periferica di rete, l'errore generato è `Errore di accesso al percorso/file.` Un esempio è il tentativo da parte di un processo di aprire un file per l'output in una directory condivisa per la sola lettura.

Per ulteriori informazioni rigurardanti l'uso dei file in rete, vedere le istruzioni LOCK e UNLOCK.

Esempi

```
10 OPEN "I",2,"INVENT"
```

Apre il file 2, INVENT, per input sequenziale.

OPEN "COM(n) (istruzione)

Funzione

Assegna un buffer per il supporto delle comunicazioni asincrone RS-232 con altri computer e unità periferiche, in modo analogo all'istruzione OPEN con i file di disco.

Sintassi

OPEN "COM[*n*]:[*velocità*][,*parità*][,*dati*] [,*stop*][,RS][,CS[*n*]][,DS[*n*]] [,CD[*n*]][,LF] [,PE][" **AS** [#]*numerofile* [LEN=*numero*]

Commenti

COM[*n*] è un'unità di comunicazioni valida: COM1: o COM2:.

velocità è un valore intero alfanumerico che specifica la velocità di trasmissione/ricezione (baud rate).

Le velocità valide sono le seguenti:

75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, e 9600. La velocità predefinita è di 300 bps (baud per secondo).

parità è un carattere letterale che specifica la parità per la trasmissione e la ricezione.

I caratteri validi per la parità sono i seguenti:

S SPACE. Il bit di parità viene sempre trasmesso e ricevuto come uno spazio (bit 0).

M MARK. Il bit di parità viene sempre trasmesso e ricevuto come un segno (bit 1).

O ODD. Parità trasmessa dispari; parità dispari verificata alla ricezione. Il valore predefinito è pari.

E EVEN. Parità trasmessa pari; parità pari verificata alla ricezione. Il valore predefinito è pari.

N NONE. Nessuna parità trasmessa e nessun controllo di ricezione.

dati è un valore alfanumerico intero che indica il numero dei bit di dati trasmessi/ricevuti.

I valori validi per i numeri di bit di dati sono 4, 5, 6, 7 ed 8, ed il valore predefinito è 7.

Avvertenza L'uso di quattro bit di dati senza parità non è ammesso, così come l'uso di otto bit di dati con una qualsiasi parità.

stop è un'espressione alfanumerica intera che restituisce un numero di file valido.

I valori validi per i bit di stop sono 1 e 2. Se il valore viene omesso, 75 e 110 bps trasmettono due bit di stop. Tutti gli altri ne trasmettono uno.

numerofile è un numero tra 1 ed il numero massimo di file permessi. Un'unità di comunicazioni può essere aperta con un solo numero di file per volta.

Il *numerofile* viene associato al file fino a che il file rimane aperto; esso viene utilizzato per riferirsi al file in altre istruzioni COM I/O.

Qualsiasi errore di codificazione nella stringa del nome del file determina l'errore `Nome di file errato`.

numero è il numero massimo di byte che può essere letto dal buffer di comunicazioni durante l'uso del valore predefinito di GET e PUT (128 byte).

L'errore `Timeout periferica` avviene se il segnale *DSR* (Data Set Ready) non viene individuato.

Le opzioni RS, CS, DS, DC, LF, e PE influiscono sui segnali di riga nel modo seguente:

Opzione	*Funzione*
RS	sopprime RTS (Request To Send)
CS[*n*]	controlla CTS (Clear To Send)
DS[*n*]	controlla DSR (Data Set Ready)
CD[*n*]	controlla CD (Carrier Detect)
LF	invia un avanzamento di riga (linefeed) per ciascun RITORNO
PE	attiva il controllo di parità

n è il numero dei millesimi di secondo (da 0 a 65535) di attesa del segnale, prima che un errore di timeout della periferica si verifichi. I valori predefiniti sono: CS1000, DS1000 e CD0. Se è stato specificato RS, la predefinizione è CS0. Se si omette *n*, il tempo d'attesa viene impostato a 0.

Per ulteriori informazioni circa le comunicazioni, si veda l'appendice F nel *Manuale dell'utente.*

Esempi

Nell'esempio seguente, il file 1 viene aperto per le comunicazioni con tutti i valori predefiniti: velocità 300 bps, parità pari, sette bit di dati ed un bit di stop.

```
10 OPEN "COM1:" AS 1
```

Nel prossimo esempio, il file 2 viene aperto per comunicazioni a 2400 bps. Per parità e numeri di bit di dati vengono utilizzati i valori predefiniti sopra descritti.

```
20 OPEN "COM1:2400" AS #2
```

In quest'ultimo esempio, il file 1 viene aperto per comunicazioni asincrone a 2400 bit/secondo. Qui, non è effettuato nessun controllo o invio di parità.

```
10 OPEN "COM1:1200,N,8" AS #1
```

OPTION BASE (istruzione)

Funzione

Dichiara il valore minimo degli indici di matrice.

Sintassi

OPTION BASE *n*

Commenti

n è 1 o 0. Il valore predefinito è 0.

Se si esegue l'istruzione OPTION BASE 1, il valore più basso possibile per un indice di matrice è 1.

Un indice di matrice non può mai avere un valore negativo.

OPTION BASE causa un errore soltanto se si cambia il valore di base. Ciò permette ai programmi concatenati di avere istruzioni OPTION BASE finché il valore non viene cambiato dall'impostazione iniziale.

Avvertenza Prima della definizione o l'uso di una matrice, bisogna codificare l'istruzione OPTION BASE. Il tentativo di cambiare il valore della base dopo aver iniziato ad usare una matrice determina un errore.

OUT (istruzione)

Funzione

Invia un byte alla porta di output di una macchina.

Sintassi

OUT *h,j*

Commenti

h e *j* sono espressioni intere. *h* deve rientrare nell'intervallo da 0 a 65535. *j* deve rientrare nell'intervallo da 0 a 255. *h* è il numero della porta della macchina, mentre *j* rappresenta il dato da trasmettere.

OUT è l'istruzione complementare alla funzione INP.

Esempi

```
100 OUT 12345,225
```

Invia il valore decimale 225 attraverso la porta numero 12345. In linguaggio d'assemblaggio, questa operazione equivale a quanto segue:

```
MOV DX,12345
MOV AL,255
OUT DX,AL
```

PAINT (istruzione)

Funzione

Riempie una figura grafica con l'attributo scelto.

Sintassi

PAINT (*x inizio,y inizio*)[,*colorazione*[,*bordo*][,*fondo*]]

Commenti

L'istruzione PAINT riempie una figura grafica con gli attributi di bordo e colorazione specificati. Se si omette *colorazione*, viene usato il valore predefinito per il colore di primo piano (3 o 1). Per *bordo* il valore predefinito è quello assegnato a *colorazione*. Per informazioni più dettagliate consultare le istruzioni COLOR e PALETTE.

PAINT deve iniziare ad un punto che non sia compreso nel bordo; altrimenti non produce nessun effetto.

PAINT è in grado di riempire qualsiasi figura. Tuttavia, la colorazione di bordi seghettati o di figure molto complesse potrebbe determinare l'errore `Memoria esaurita`. Per aumentare la quantità di spazio disponibile nello stack si può usare l'istruzione CLEAR.

I punti specificati oltre i limiti dello schermo non vengono tracciati e non si verifica alcun errore.

Per una descrizione delle diverse modalità di schermo, si veda l'istruzione SCREEN.

Colorazione a motivo

La colorazione a motivo con PAINT è simile alla stilizzazione di LINE. Analogamente a LINE, ogni volta che un punto viene tracciato sullo schermo, PAINT fa riferimento ad una maschera (o matrice) rappresentante un motivo.

Se si omette *colorazione*, viene usato il valore predefinito per il colore di primo piano (3 o 1).

Se *colorazione* è una formula numerica, il numero deve rappresentare un colore valido, che viene poi usato per colorare l'area come in precedenza.

Se *colorazione* è una formula alfanumerica, la colorazione viene eseguita nel seguente modo.

La matrice del motivo è sempre larga otto bit e può variare da 1 a 64 byte in lunghezza. Nel corso dell'inserimento di punti, ciascun byte della stringa del motivo maschera otto bit lungo l'asse x. Inoltre ciascun byte della stringa viene adeguatamente ruotato per l'allineamento lungo l'asse y, in modo che:

byte_maschera_motivo = y MOD lunghezza_motivo

dove *y* è la posizione del cursore grafico sull'asse y.

lunghezza_motivo è la lunghezza in byte della stringa del motivo definita dall'utente (da 1 a 64 byte).

Ciò viene fatto in modo che la struttura del motivo venga ripetuta uniformemente sull'intero schermo (come se fosse utilizzato PAINT (0,0)..).

```
x Aumenta --> Bit di motivo
x,y      8 7 6 5 4 3 2 1
0,0     |x|x|x|x|x|x|x|x|   Byte di motivo 1
0,1     |x|x|x|x|x|x|x|x|   Byte di motivo 2
0,2     |x|x|x|x|x|x|x|x|   Byte di motivo 3
.
.
.
0,63    |x|x|x|x|x|x|x|x|   Byte di motivo 64
                            (massimo permesso)
```

In alta risoluzione (SCREEN 2), lo schermo può essere colorato con delle X grazie all'istruzione seguente:

```
PAINT
(320,100),CHR$(&H81)+CHR$(&H42)+CHR$(&H24)+CHR$(&H18)+
CHR$(&H18)+CHR$(&H24)+CHR$(&H81)
```

Sullo schermo appare il seguente motivo:

```
x aumenta -->
0,0   |x| | | | | | |x|   CHR$(&H81)   Byte di motivo 1
0,1   | |x| | | | |x| |   CHR$(&H42)   Byte di motivo 2
0,2   | | |x| | |x| | |   CHR$(&H24)   Byte di motivo 3
0,3   | | | |x|x| | | |   CHR$(&H18)   Byte di motivo 4
0,4   | | | |x|x| | | |   CHR$(&H18)   Byte di motivo 5
0,5   | | |x| | |x| | |   CHR$(&H24)   Byte di motivo 6
0,6   | |x| | | | |x| |   CHR$(&H42)   Byte di motivo 7
0,6   |x| | | | | | |x|   CHR$(&H81)   Byte di motivo 8
```

Dal momento che in risoluzione media (SCREEN 1) vi sono due bit per pixel, ciascun byte della struttura del motivo descrive solo quattro pixel. In questo caso ogni due bit del byte del motivo descrivono uno dei quattro colori possibili associati con ciascuno dei quattro pixel da inserire.

fondo specifica la struttura del motivo di fondo o il byte di colore da saltare durante la ricerca dei bordi. *fondo* è una stringa che restituisce un carattere. Il colore predefinito è CHR$(0).

Occasionalmente si potrebbe voler colorare con un motivo un'area già colorata che è dello stesso colore di due righe consecutive nella struttura del motivo. Quando incontra due righe consecutive dello stesso colore del punto in fase di inserimento, PAINT si arresta (il punto viene evidenziato). Non è possibile tracciare righe blu e rosse alternate su un fondo rosso senza utilizzare *fondo*. Non appena il primo pixel rosso viene tracciato PAINT si arresta. La specificazione del rosso (CHR$(&HAA)) come attributo di fondo, permette la traccia della riga rossa sul fondo rosso.

Nella stringa di motivo non possono essere specificati più di due byte che corrispondono all'attributo di fondo. Ciò determina infatti l'errore `Chiamata di funzione illegale`.

Esempi

```
10 CLS
20 SCREEN 1
30 LINE (0,0)-(100,150),2,B
40 PAINT (50,50),1,2
50 LOCATE 20,1
```

L'istruzione PAINT alla riga 40 riempie col colore 1 il quadrato tracciato alla riga 30.

PALETTE, PALETTE USING (istruzioni)

Funzione

Cambia uno o più colori nel set a disposizione.

Sintassi

PALETTE [*attributo,colore*]
PALETTE USING *nome-matrice-intera* (*indice-matrice*)

Commenti

L'istruzione PALETTE funziona soltanto con i sistemi che dispongono di un adattatore grafico EGA IBM. Un set di colori di GW-BASIC contiene un gruppo di colori in cui ogni colore è individuato da un *attributo*. Ciascun *attributo* è abbinato ad un effettivo *colore* di visualizzazione. Questo colore dipende dall'impostazione della modalità di schermo e dall'hardware di cui si dispone.

PALETTE senza argomenti imposta il set di colori ad un'impostazione iniziale conosciuta. Questa impostazione è la stessa che si ha quando i colori vengono inizializzati.

Se vengono specificati degli argomenti, *colore* viene visualizzato ogni volta che l'*attributo* viene specificato, in qualsiasi istruzione in cui viene specificato un colore. Qualsiasi cambio di colore sullo schermo viene eseguito immediatamente. Occorre notare che quando le istruzioni grafiche usano argomenti di colore, fanno riferimento in effetti ad attributi anziché a colori. PALETTE abbina poi gli attributi ad effettivi colori.

Ad esempio, si supponga che il set di colori corrente contenga i colori 0, 1, 2 e 3. L'istruzione DRAW che segue:

```
DRAW "C3L100"
```

seleziona l'attributo 3, e traccia una riga di 100 pixel utilizzando il colore associato all'attributo 3 (che in questo caso è il colore 3).

Se viene eseguita poi l'istruzione seguente:

```
PALETTE 3,2
```

il colore associato all'attributo 3 diviene il numero 2. Di conseguenza, tutti i testi o grafici che usano l'attributo 3 e sono correntemente visualizzati sullo schermo cambiano immediatamente nel colore 2. Anche tutti i testi o grafici che verranno visualizzati successivamente con l'attributo 3 verranno presentati nel colore 2. Quindi il nuovo set di colori contiene 0, 1, 2 e 2.

Con l'opzione USING, tutti gli inserimenti effettuati nel set possono essere modificati con un'istruzione PALETTE. L'argomento *nome-matrice-intera* è il nome della matrice intera, mentre *indice-matrice* specifica l'indice del primo elemento nel *nome-matrice-intera* da usare per l'impostazione del set di colori. A ciascun *attributo* nel set di colori viene assegnato un colore corrispondente dalla matrice intera. La matrice deve essere di dimensione sufficiente ad impostare tutte le voci del set di colori dopo *indice-matrice*. Ad esempio, se si assegnano colori a tutti i 16 attributi, e l'indice del primo elemento della matrice fornito nell'istruzione PALETTE USING è 5, la matrice deve essere di dimensioni tali da contenere almeno 20 elementi (il numero degli elementi da 5 a 20 è 16):

```
DIM PAL%(20)
.
.
.
PALETTE USING PAL%(5)
```

Se l'argomento *colore* in un elemento della matrice è -1, l'associazione all'*attributo* non cambia. Tutti gli altri valori negativi non sono validi per *colore*.

L'argomento di colore può essere usato con l'istruzione COLOR per impostare un colore di testo predefinito (si ricordi che gli argomenti di colore in altre istruzioni di BASIC sono in effetti equivalenti agli *attributi* usati in questa discussione). Questo argomento di colore specifica come i caratteri di testo appaiono sullo schermo. Sotto un'impostazione di set di colori iniziale comune, i punti colorati con *attributo* 0 appaiono sullo schermo in nero. Usando l'istruzione PALETTE, si potrebbe cambiare l'*attributo* 0 da nero a bianco.

Ricordarsi che un'istruzione PALETTE eseguita senza parametri assegna a tutti gli attributi il loro colore predefinito.

La tabella seguente elenca i vari tipi di *attributo* e *colore* per i vari tipi di monitor e modalità di schermo:

Tabella 1 Tipi di attributi e di colori di schermo

Modalità di schermo	*Monitor collegato*	*Adattatore*	*Gamma attributi*	*Gamma colori*
0	Monocolore	MDPA	NA	NA
	Monocolore	EGA	0-15	0-2
	A colori	CGA	NA	0-31[a]
	A colori/Potenziato[d]	EGA	0-31[a]	0-15
1	A colori	CGA	NA	0-3
	A colori/Potenziato[d]	EGA	0-3	0-15
2	A colori	CGA	NA	0-1
	A colori/Potenziato[d]	EGA	0-1	0-15
7	A colori/Potenziato[d]	EGA	0-15	0-15
8	A colori/Potenziato[d]	EGA	0-15	0-15
9	Potenziato[d]	EGA[b]	0-3	0-15
	Potenziato[d]	EGA[c]	0-15	0-63
10	Monocolore	EGA	0-3	0-8

a Gli attributi da 16 a 31 si riferiscono a versioni intermittenti dei colori da 0 a 15

b EGA con 64K di memoria

c EGA con memoria maggiore di 64K

d Enhanced Color Display IBM

NA = Non Applicabile

CGA = IBM Color Graphics Adapter

EGA = IBM Enhanced Graphics Adapter

MDPA = IBM Monochrome Display and Printer Adapter

Per un elenco dei colori disponibili nelle varie modalità di schermo, monitor, e combinazioni di adattatori grafici, vedere la pagina di riferimento dell'istruzione SCREEN.

Esempi

```
PALETTE 0,2               'Cambia tutti i punti colorati
                          'con attributo 0 nel colore 2
PALETTE 0,-1              'Non modifica il set di colori
PALETTE USING A%(0)       'Cambia ogni voce del set di
                          'colori. Dal momento che la
                          'matrice viene inizializzata
                          'con il valore 0, tutti gli
                          'attributi sono impostati per
                          'visualizzare il colore zero.
                          'Lo schermo appare in un
                          'singolo colore. E' tuttavia
                          'possibile eseguire istruzioni
                          'di BASIC.
PALETTE                   'Imposta ciascuna voce del set
                          'di colori al colore di
                          'visualizzazione iniziale
                          'appropriato. I colori iniziali
                          'dipendono dalla configurazione
                          'dell'hardware.
```

PCOPY (comando)

Funzione

Copia una pagina di schermo su un'altra in tutte le modalità di schermo.

Sintassi

PCOPY *pagina d'origine, pagina di destinazione*

Commenti

La *pagina d'origine* è un'espressione intera nell'intervallo da 0 a *n*, in cui *n* dipende dalla memoria del monitor e dalla dimensione per pagina della modalità di schermo corrente.

pagina di destinazione ha le stesse caratteristiche di *pagina d'origine*.

Per informazioni più dettagliate, vedere CLEAR e SCREEN.

Esempi

Il comando seguente copia i contenuti di pagina 1 su pagina 2:

```
PCOPY 1,2
```

PEEK (funzione)

Funzione

Esegue la lettura da una posizione in memoria specifica.

Sintassi

PEEK(*a*)

Commenti

Restituisce il byte (valore decimale intero nell'intervallo da 0 a 255) letto alla posizione di memoria specificata (*a*). Il valore di *a* deve rientrare nell'intervallo da 0 a 65535.

L'ultima istruzione DEF SEG eseguita determina l'indirizzo assoluto al quale verrà eseguita la lettura.

PEEK è la funzione complementare all'istruzione POKE.

Esempi

```
10 A=PEEK(&H5A00)
```

Il valore del byte, memorizzato nella posizione di memoria esadecimale 5A00 (23040 decimale) assegnata dall'utente, verrà associato alla variabile A.

PEN (funzione e istruzione)

Funzione

Esegue la lettura delle coordinate della penna luminosa.

Sintassi

Come istruzione:

PEN ON
PEN OFF
PEN STOP

Come funzione:

x = **P**(*n*)

Commenti

x è la variabile numerica che riceve il valore di PEN.

n è un valore intero nell'intervallo da 0 a 9.

PEN ON attiva la funzione per la lettura di PEN.

PEN OFF disattiva la funzione per la lettura di PEN.

PEN STOP disattiva il trapping. Memorizza l'evento in modo che all'esecuzione di PEN ON avvenga un trapping immediato.

x = PEN(*n*) legge le coordinate della penna luminosa.

La posizione iniziale di PEN è OFF. Prima che qualsiasi chiamata di funzione per la lettura di PEN possa essere effettuata, bisogna eseguire l'istruzione PEN ON. In caso contrario, si verifica l'errore `Chiamata di funzione illegale`.

Coordinate della penna luminosa:

n = 0 Se PEN è disattivato sin dall'ultimo campionamento, viene restituito -1; altrimenti viene restituito 0.

n = 1 Restituisce la coordinata x (in pixxel) del punto in cui PEN è stato attivato per l'ultima volta. Il valore è compreso tra 0 e 319 in risoluzione media e tra 0 e 639 in alta risoluzione.

n = 2 Restituisce la coordinata y (in pixel) del punto in cui PEN è stato attivato per l'ultima volta. Il valore può variare da 0 a 199.

n = 3 Restituisce il valore del parametro di PEN corrente. Se PEN è attivo viene restituito -1, altrimenti viene restituito 0.

n = 4 Restituisce l'ultima coordinata x (in pixel) conosciuta. Il valore è compreso tra 0 e 319 in risoluzione media, e tra 0 e 639 in alta risoluzione.

n = 5 Restituisce l'ultima coordinata y (in pixel) conosciuta. Il valore è compreso tra 0 e 319 in risoluzione media, e tra 0 e 639 in alta risoluzione.

n = 6 Restituisce il numero di riga in cui PEN è stato attivato per l'ultima volta. Il numero è compreso tra 1 e 24.

n = 7 Restituisce il numero di colonna in cui PEN è stato attivato per l'ultima volta. Il numero è compreso tra 1 e 40, o tra 1 e 80, in relazione alla larghezza dello schermo.

n = 8 Restituisce il numero dell'ultima riga valida in cui PEN è stato attivato. Il numero è compreso tra 1 e 24.

n = 9 Restituisce il numero dell'ultima colonna valida in cui PEN è stato attivato. Il numero è compreso tra 1 e 40, o tra 1 e 80, in relazione alla larghezza dello schermo.

Per migliorare la rapidità di esecuzione, disattivare la penna usando l'istruzione PEN OFF per i programmi che non ne fanno uso.

Se la penna si trova nella zona di bordo dello schermo, i valori restituiti non saranno esatti.

Esempi

```
50 PEN ON
60 FOR I=1 to 500
70 X=PEN(0):X1=PEN(3)
80 Print X,X1
90 NEXT
100 PEN OFF
```

Questo esempio stampa il valore della penna all'atto dell'ultimo campionamento, e il valore corrente.

PLAY (istruzione)

Funzione

Esegue il suono di musica includendo un linguaggio musicale macro in una stringa di caratteri.

Sintassi

PLAY *stringa*

Commenti

I comandi PLAY sono i seguenti:

A-G [#,+,-]	A-G sono le note. # o + producono un diesis; - produce un bemolle. Qualsiasi nota seguita da #, +, o da - deve riferirsi ad un tasto nero su un piano.
L(*n*)	Imposta la lunghezza di ciascuna nota. L4 è un quarto di nota, L1 è una nota intera, e così via. *n* può variare da 1 a 64. La lunghezza può anche seguire una nota per determinarne il cambio di lunghezza.
MF	Musica di primo piano. Le istruzioni PLAY e SOUND vengono eseguite in primo piano. Cioè, ciascuna nota o suono successivo non viene attivato fino a che la nota o suono precedente è terminato. E' l'assunzione iniziale.
MB	Musica di sottofondo. Le istruzioni PLAY e SOUND vengono eseguite in sottofondo. Cioè, ciascuna nota o suono viene posto nel buffer in modo di permettere al programma di continuare l'esecuzione mentre la musica suona in sottofondo. E' possibile suonare fino a 32 note in sottofondo simultaneamente.
MN	Musica normale. Ciascuna nota suona sette ottavi del tempo determinato da L (lunghezza).

ML	Legato. Ciascuna nota suona per l'intero tempo impostato da L.
MS	Staccato. Ciascuna nota suona tre quarti del tempo determinato da L.
N(*n*)	Suona la nota *n*. *n* può variare da 0 a 84. Nei 7 ottavi possibili, vi sono 84 note. *n* uguale a 0 indica una pausa.
O(*n*)	L'ottava 0 inizializza l'ottava corrente. Vi sono 7 ottave (da 0 a 6). Il valore predefinito è 4. Do3 è all'inizio dell'ottava 3.
P(*n*)	Pausa. Può variare da 1 a 64.
T(*n*)	Tempo. Imposta il numero di L4 in un minuto. *n* può variare da 32 a 255. Il valore predefinito è 120.
.(punto)	Un punto dopo una nota aumenta la durata del suono della nota dei 3/2 per L (lunghezza della nota) per T (tempo). Se dopo una nota appaiono più punti, la scala del suono viene modificata conseguentemente. Ad esempio, A estende la durata del suono della nota ad una volta e mezzo il tempo impostato da L (lunghezza della nota) per T (tempo); due punti posti dopo A (A..) estendono la durata del suono della nota a 9/4 volte il valore originale; una A seguita da tre punti (A...) estende la durata a 27/8 volte, ecc.. Inoltre i punti possono figurare dopo una P (pausa) al fine di aumentare la durata della pausa nella misura sopra descritta.
X*stringa*;	Esegue una sottostringa. *stringa* è una variabile assegnata a una stringa di comandi PLAY. A causa della lenta esecuzione dell'interruzione dell'orologio, alcune note non suonano a ritmi alti; ad esempio, 1.64 a T255. Questa combinazione di nota/tempo deve essere impostata per tentativi.
>*n*	Il simbolo (>) posto prima della nota *n* suona quella nota nella prima ottava più alta che segue.
<*n*	Il simbolo (<) posto prima della nota *n* suona quella nota nella prima ottava più bassa che segue.

Avvertenza Gli argomenti numerici seguono la stessa sintassi descritta per l'istruzione DRAW.

n come argomento può essere una costante o una variabile con il segno uguale (=) posto davanti (=*variabile*). Dopo la variabile è necessaria l'inclusione di un punto e virgola (lo stesso vale per la variabile X*stringa*).

PLAY(n) (funzione)

Funzione

Restituisce il numero delle note correntemente nella coda della musica di fondo.

Sintassi

PLAY(*n*)

Commenti

n è un argomento fittizio e può essere di qualsiasi valore.

In modalità di musica di primo piano PLAY(*n*) restituisce 0.

Il valore massimo di x restituito è 32.

Esempi

```
10 ' quando nella coda rimangono
20 ' 4 note suona un altro tono
30 PLAY "MBABCDABCDABCD"
40 IF PLAY (0) =4 then 200.
.
.
200 PLAY "MBCDEFCDEF"
```

PMAP (funzioni per grafici)

Funzione

Associa delle espressioni a delle coordinate logiche o fisiche.

Sintassi

x=**PMAP** (*espressione,funzione*)

Commenti

Questa funzione è valida soltanto per modalità grafiche.

x è la coordinata fisica del punto da associare.

espressione è una variabile o un'espressione numerica.

Funzione	*Associazione*
0	espressioni logiche e x fisica
1	espressioni logiche e y fisica
2	espressioni fisiche e x logica
3	espressioni fisiche e y logica

PMAP viene usato in combinazione con WINDOW e VIEW per convertire le coordinate.

POINT (funzione)

Funzione

Legge il valore dell'attributo o del colore di un pixel sullo schermo.

Sintassi

POINT(*x,y*)
POINT(*funzione*)

Commenti

Nella prima sintassi, *x* e *y* rappresentano le coordinate del punto da e saminare.

Se il punto fornito non rientra nei limiti dello schermo, viene restituito il valore -1.

Per i numeri di colore e di attributo validi, si vedano le istruzioni COLOR e PALETTE.

POINT in combinazione con un solo argomento permette di leggere le coordinate grafiche correnti.

POINT(*funzione*) restituisce i valori delle coordinate grafiche x e y correnti nel seguente modo:

Funzione	*Restituisce*
0	la coordinata fisica corrente di x.
1	la coordinata fisica corrente di y.
2	la coordinata logica corrente di x, se WINDOW è attiva; altrimenti, la coordinata fisica corrente di x come in 0.
3	la coordinata logica corrente di y, se WINDOW è attiva; altrimenti, la coordinata fisica corrente di y come in 1.

Esempio 1

```
10 SCREEN 1
20 FOR C=0 TO 3
30 PSET (10,10),C
40 IF POINT(10,10) <>C THEN PRINT "BASIC GUASTO!"
50 NEXT C
RUN
```

Esempio 2

Quanto segue inverte lo stato corrente di un punto:

```
10 SCREEN 2
20 IF POINT(I,I) <>0 THEN PRESET(I,I) ELSE PSET(I,I)
RUN
```

Esempio 3

Il seguente è un altro modo per invertire un punto:

```
20 PSET (I,I),1-POINT(I,I)
RUN
```

POKE (istruzione)

Funzione

Scrive un byte di dati in una posizione di memoria.

Sintassi

POKE *a,b*

Commenti

a e *b* sono espressioni intere.

L'espressione intera *a* è l'indirizzo della posizione in memoria nella quale viene eseguita la scrittura del byte. L'ultima istruzione DEF SEG eseguita determina l'indirizzo. GW-BASIC non esegue il controllo di alcun indirizzo specificato.

L'espressione intera *b* è il dato da scrivere.

b deve rientrare nell'intervallo da 0 a 255. *a* deve rientrare nell'intervallo da 0 a 65535.

La funzione POKE è complementare a PEEK. L'argomento di PEEK è l'indirizzo dal quale viene letto un byte.

POKE e PEEK aiutano a memorizzare efficacemente i dati, per caricare e per inviare e ricevere argomenti e risultati dalle subroutine in linguaggio d'assemblaggio.

Esempi

```
20 POKE &H5A00,&HFF
```

Pone il valore decimale 255 (&HFF) nella posizione dell'offset esadecimale (23040 decimale). Si veda l'esempio della funzione PEEK.

POS (funzione)

Funzione

Indica la posizione corrente del cursore.

Sintassi

POS(c)

Commenti

La posizione più a sinistra è 1.

c è un argomento fittizio.

Esempi

```
10 CLS
20 WIDTH 80
30 A$ = INKEY&:IF A$="" THEN GOTO 30 ELSE PRINT A$;
40 IF POS(X)#10 THEN PRINT CHR$ (13);
50 GOTO 30
```

Determina il ritorno di riga (carriage return) dopo la stampa del 10mo carattere su ciascuna riga dello schermo.

PRESET e PSET (istruzioni)

Funzione

Visualizzano un punto specificato dello schermo in modalità grafica.

Sintassi

PRESET(*x,y*)[,*colore*]
PSET(*x,y*)[,*colore*]

Commenti

(*x,y*) sono le coordinate del punto.

colore è il colore del punto.

Le coordinate possono essere indicate in forma assoluta o relativa.

Forma assoluta

E' più comune. Indica direttamente un punto senza tener conto dell'ultimo punto al quale si è fatto riferimento. Ad esempio:

```
(10,10)
```

Forma relativa

Si presenta nella forma STEP (*offset x,offset y*). Indica un punto in relazione all'ultimo punto a cui si è fatto riferimento. Ad esempio:

```
STEP(10,10)
```

Le coordinate del punto possono oltrepassare il bordo dello schermo. Tuttavia, i valori che non rientrano nell'intervallo intero da -32768 a 32767 determinano l'errore `Valore troppo elevato`.

Sia in risoluzione alta che in risoluzione media, (0,0) è l'angolo superiore sinistro, mentre (0,199) è l'angolo inferiore sinistro.

Per informazioni più dettagliate, si vedano le istruzioni COLOR e PALETTE.

Se il valore specificato per *colore* è maggiore di 3, si verifica l'errore `Chiamata di funzione illegale.`

Esempio 1

L'esempio seguente traccia una diagonale dal punto (0,0) al punto (100,100).

```
10 CLS
20 SCREEN 1
30 FOR I=A TO 100
40 PSET (I,I)
50 NEXT
60 LOCATE 14,1
```

Esempio 2

L'esempio seguente cancella la riga impostando ogni pixel a 0.

```
40 FOR I=100 TO 0 STEP -1
50 PSET(I,I),0
60 NEXT I
```

PRINT (istruzione)

Funzione

Visualizza dei risultati sullo schermo.

Sintassi

PRINT [*elencoespressioni*][;]
?[*elencoespressioni*][;]

Commenti

Se si omette *elencoespressioni*, viene visualizzata una riga vuota.

Se invece viene incluso *elencoespressioni*, vengono visualizzati i valori delle espressioni. Le espressioni nell'elenco possono essere numeriche e/o alfanumeriche, separate da virgole, spazi o punti e virgola. Le costanti alfanumeriche nell'elenco devono essere racchiuse tra virgolette.

Per informazioni più dettagliate circa le stringhe, vedere la funzione STRING$.

Se si usa l'editor di GW-BASIC, si può utilizzare un punto interrogativo (?) al posto della parola PRINT.

Le posizioni di stampa

GW-BASIC divide la riga in zone di stampa da 14 spazi l'una. La posizione di ciascun elemento stampato viene determinata dalla punteggiatura usata per separare gli elementi nell'elenco:

Separatore	*Posizione di stampa*
,	Inizio zona seguente
;	Immediatamente dopo l'ultimo valore
spazi(o)	Immediatamente dopo l'ultimo valore

Se un elenco di espressioni viene chiuso da una virgola, un punto e virgola, o la funzione SPC o TAB, l'istruzione PRINT seguente inizia la stampa sulla stessa riga, con gli spazi adeguati. Se invece un elenco di espressioni non viene chiuso da una virgola, un punto e virgola, o la funzione SPC o TAB, viene posto un ritorno di riga (carriage return) alla fine di ogni riga (GW-BASIC pone cioè il cursore all'inizio della riga seguente).

Al riempimento dello schermo in larghezza, viene automaticamente inserito un avanzamento/ritorno di riga. La larghezza è di 40 o 80 caratteri, a seconda dell'impostazione dell'istruzione WIDTH. Ciò determina il salto di una riga quando vengono stampati esattamente 40 (o 80) caratteri, sempre che l'istruzione PRINT non termini con un punto e virgola.

I numeri stampati sullo schermo sono sempre seguiti da uno spazio. I numeri positivi sono preceduti da uno spazio. I numeri negativi sono preceduti da un meno (-). I numeri a precisione singola sono rappresentati da sette (o meno) cifre in un formato intero o a punto fisso.

Per informazioni riguardanti l'invio di dati alla stampante, si vedano le istruzioni LPRINT e LPRINT USING.

Esempi

```
10 X$= STRINGA$(10,45)
20 PRINT X$"RELAZIONE MENSILE" X$
----------RELAZIONE MENSILE----------
Ok
```

45 è l'equivalente decimale del simbolo ASCII per il meno (-).

PRINT USING

Funzione

Stampa delle stringhe o dei numeri usando il formato specificato.

Sintassi

PRINT USING *stringa;elencoespressioni*[;]

Commenti

stringa è una stringa variabile o alfanumerica composta da caratteri di formattazione speciale. I caratteri di formattazione determinano il campo ed il formato delle stringhe o dei numeri stampati.

elencoespressioni è composto da espressioni alfanumeriche o numeriche separate da punti e virgola.

Campi alfanumerici

I tre caratteri sotto elencati possono essere usati per la formattazione di un campo alfanumerico:

!	Specifica che bisogna stampare soltanto il primo carattere della stringa.
n spazi\\	Specifica che bisogna stampare 2+*n* caratteri della stringa. Se i backslash (\\) vengono digitati senza spazi, vengono stampati due caratteri; se vengono digitati con uno spazio, vengono stampati tre caratteri, e così via. Se la stringa è più lunga del campo, i caratteri in eccedenza vengono ignorati. Se invece il campo è più lungo della stringa, la stringa viene allineata a sinistra e la parte destra del campo viene completata da spazi.

Ad esempio:

```
10 A$="FARE":B$="ATTENZIONE"
30 PRINT USING "!";A$;B$
40 PRINT USING"\  \";A$;B$
50 PRINT USING"\        \";A$;B$;"!!"
RUN
FA
FAREATTE
FARE      ATTENZIONE!!
```

& Specifica una campo alfanumerico a lunghezza variabile. Quando il campo viene specificato con &, la stringa viene stampata esattamente nel modo in cui è stata inserita. Ad esempio:

```
10 A$="FARE":B$="ATTENZIONE"
20 PRINT USING "!";A$
30 PRINT USING "&";B$
RUN
FATTENZIONE
```

Campi numerici

I caratteri speciali che seguono possono essere usati per formattare i campi numerici:

Questo carattere viene usato per rappresentare la posizione di ciascuna cifra. Se il numero da stampare è composto da meno cifre di quante sono le posizioni specificate, il numero viene allineato a destra e la parte sinistra del campo viene completata da spazi.

E' possibile inserire un punto decimale in qualsiasi posizione del campo. Se il formato della stringa specifica che una cifra deve precedere un punto decimale, quella cifra viene sempre stampata (come 0, se necessario). I numeri vengono arrotondati come necessario. Ad esempio:

```
PRINT USING "##.##";.78
0.78

PRINT USING "###.##";987.654
987.65
Ok
```

```
PRINT USING "##.##"   ;10.2,5.3,66.789,.234
10.20    5.30    66.79    0.23
```

Nell'ultimo esempio, sono stati inseriti tre spazi alla fine della stringa di formato per separare i valori stampati sulla riga.

\+ Il segno più posto all'inizio o alla fine della stringa di formato fa in modo che il segno del numero (più o meno) venga stampato rispettivamente prima o dopo il numero.

\- Il segno meno alla fine del campo di formato determina la stampa dei numeri negativi con il segno meno all'inizio. Ad esempio:

```
PRINT USING"+##.##";-68.95,2.4,55.6,-9
-68.95 +2.40 +55.60 -0.90
OK

PRINT USING"##.##-";-68.95,22.449,-7.01
68.95 22.45 7.01-
OK
```

** Un doppio asterisco all'inizio della stringa di formato determina la sostituzione degli spazi con asterischi. Il doppio asterisco può anche specificare due posizioni numeriche in più. Ad esempio:

```
PRINT USING "**#.#";12.39,-0.9,765.1
*12.4*  -09765.1
Ok
```

$$ Un doppio simbolo del dollaro posto all'inizio della stringa di formato, determina la stampa del simbolo all'immediata sinistra del numero formattato. Inoltre il doppio simbolo del dollaro può indicare due posizioni numeriche in più, una delle quali è lo stesso simbolo. Con $$ il formato esponenziale non può essere usato. I numeri negativi non non possono essere usati, a meno che abbiano il segno meno (-) a destra, .

```
PRINT USING "$$###.##";456.78
$456,78
```

**$ Questo simbolo posto all'inizio della stringa di formato combina gli effetti dei due simboli precedenti. Gli spazi vengono riempiti da asterischi, ed il simbolo viene stampato prima del numero. **$ specifica tre posizioni numeriche in più, una delle quali è il simbolo del dollaro. Ad esempio:

```
PRINT USING "**$##.##";2,34
***$2,34
```

Una virgola a sinistra del punto decimale nella stringa di formato, determina la stampa di una virgola sulla sinistra di ogni terza cifra alla sinistra del punto decimale (cioè l'inserimento di un separatore per le migliaia). Una virgola posta alla fine della stringa di formato, viene stampata come parte della stringa.

```
PRINT USING "####.##";1234.5
1234.50
Ok
PRINT USING ",####.##";1234.5
1,234.50
Ok
```

^^^^ Quattro accenti circonflessi possono essere inseriti al termine della stringa di formato, per specificare il formato esponenziale. I quattro accenti circonflessi creano spazio per la stampa di E+xx. E' possibile specificare qualsiasi posizione per il punto decimale. Le cifre significative vengono allineate a sinistra, mentre l'esponente viene aggiustato. A meno che non si sia specificato un segno più (+) o un segno meno (-), una delle posizioni alla sinistra del punto decimale viene utilizzata per l'inserimento di uno spazio o di un segno meno (-). Ad esempio:

```
PRINT USING "##.##^^^^";234.56
2.35E+02
Ok

PRINT USING ".####^^^^-";888888
Ok

PRINT USING "+.##^^^^";123
+.12E+03
Ok
```

La virgola non viene usata come delimitatore con il formato esponenziale.

_ Un carattere di sottolineatura nella stringa di formato determina la visualizzazione del carattere letterale seguente. Ad esempio:

```
PRINT USING "_!##.##_!"'12.34
!12.34!
Ok
```

% Se il numero da stampare eccede il campo numerico specificato, viene stampato un segno di percentuale davanti al numero. Un segno di percentuale viene anche stampato davanti al numero arrotondato, quando l'arrotondamento lo rende di lunghezza maggiore del campo. Ad esempio:

```
PRINT USING "##.##";111.22
%111.22

PRINT USING ".##"';.999
%1.00
```

Se il numero di cifre specificate supera 24, si verifica l'errore `Chiamata di funzione illegale.`

PRINT# e PRINT# USING (istruzioni)

Funzione

Scrivono dei dati su un file di disco ad accesso sequenziale.

Sintassi

PRINT#*numerofile*,[USING*stringa*;]*elencoespressioni*

Commenti

numerofile è il numero usato per l'apertura del file per l'output.

stringa è formato dai caratteri di formattazione descritti nell'istruzione PRINT USING.

elencoespressioni è composto dalle espressioni numeriche e/o alfanumeriche da scrivere sul file.

Come delimitatori per le espressioni numeriche e/o alfanumeriche vengono usate le doppie virgolette. La prima virgoletta apre la riga per l'input e la seconda la chiude.

Se delle espressioni numeriche o alfanumeriche devono essere stampate non appena inserite, vanno racchiuse tra virgolette. Se le virgolette vengono omesse, il valore assegnato all'espressione numerica o alfanumerica viene stampato. Se non è stato assegnato nessun valore, il valore predefinito è 0. Le virgolette non vengono visualizzate sullo schermo. Ad esempio:

```
10 PRINT#1,A
0

10 A=26
20 PRINT#1,A
26

10 A=26
20 PRINT#1,"A"
A
```

Se in una stringa è necessaria l'inclusione di virgolette, usare CHR$(34) (il carattere ASCII equivalente alle virgolette). Ad esempio:

```
100 PRINT#1,"Ha detto "Ciao", mi sembra"
Ha detto 0, mi sembra
```

in quanto la macchina assegna il valore 0 alla variabile "Ciao".

```
100 PRINT#1, "Ha detto"CHR$(34)"Ciao"CHR$(34)", mi
sembra."
Ha detto "Ciao", mi sembra
```

Virgole, punti e virgola o spazi significativi compresi nella stringa devono essere racchiusi tra virgolette. L'esempio seguente inserisce "CALCOLATRICE AUTOMATICA" in A$, e "93604-1" in B$:

```
10 A$="CALCOLATRICE AUTOMATICA":B$="93604-1"
20 PRINT#1,A$;B$
30 INPUT#1,A$,B$
```

Per separare queste stringhe in modo corretto, scrivere virgolette successive usando CHR$(34). Ad esempio:

```
40 PRINT#1,CHR$(34);A$;CHR$(34);CHR$(34);B$; CHR$(34)
"CALCOLATRICE AUTOMATICA""93604-1"
```

L'istruzione PRINT# può anche essere usata in combinazione con l'opzione USING per controllare il formato del file su disco. Ad esempio:

```
PRINT#1,USING"$$###.##.";J;K;L
```

PRINT# non comprime i dati nel dischetto. Un'immagine dei dati viene scritta sul dischetto, in modo analogo a quello in cui verrebbe visualizzata sullo schermo con l'istruzione PRINT. Per cui, assicurarsi di delimitare i dati sul dischetto in modo che essi vengano inseriti correttamente dal dischetto.

In *elencoespressioni*, le espressioni numeriche devono essere delimitate da punti e virgola. Ad esempio:

```
PRINT#1,A;B;C;X;Y;Z
```

Se vengono usate virgole come delimitatori, gli spazi in più inseriti tra i campi di stampa vengono scritti sul dischetto. Le virgole non hanno tuttavia effetto se usate con il formato esponenziale.

Le espressioni alfanumeriche nell'elenco devono essere separate da punti e virgola. Per eseguire correttamente la formattazione delle espressioni alfanumeriche sul dischetto, usare delimitatori espliciti in *elencoespressioni*. Ad esempio, quanto segue:

```
10 A$="CALCOLATRICE":B$="93604-1"
20 PRINT#1,A$,B$
```

scrive sul dischetto:

```
CALCOLATRICE93604-1
```

Dal momento che non vi sono delimitatori, questa espressione non verrebbe inserita come due stringhe separate. Per porre rimedio al problema, inserire nell'istruzione PRINT# delimitatori espliciti, nel seguente modo:

```
30 PRINT#1,A$;",";B$
```

Questo scrive sul dischetto la seguente stringa, che può essere letta poi in due variabili alfanumeriche distinte:

```
CALCOLATRICE,93604-1
```

PUT (istruzione per file)

Funzione

Scrive un record da un buffer casuale ad un file di disco casuale.

Sintassi

PUT[#]*numerofile[,numerorecord]*

Commenti

numerofile è il numero sotto il quale il file è stato aperto.

numerorecord è il numero del record. Se il numero non viene specificato, viene assegnato al record il primo numero disponibile (dopo l'ultima istruzione PUT).

Il numero di record più alto utilizzabile è 2^{32} -1. Ciò permette di avere grossi file con record di lunghezza ridotta. Il numero di record più basso è 1.

Le istruzioni PRINT#, PRINT# USING, LSET, RSET e WRITE# possono essere usate per l'inserimento di caratteri nel buffer del file prima dell'istruzione PUT.

Nel caso di WRITE#, GW-BASIC completa il buffer con spazi, se necessario.

Qualsiasi tentativo di lettura o di scrittura oltre la fine del buffer determina l'errore `Valore troppo elevato.`

L'istruzione PUT può essere usata per i file di comunicazioni. In questo caso *numerorecord* è il numero di byte scritti sul file. *numerorecord* deve essere inferiore o uguale alla lunghezza del buffer impostato nell'istruzione OPEN"COM(n).

PUT (istruzione per grafici)

Funzione

Trasferisce immagini grafiche allo schermo.

Sintassi

PUT(*x,y*),*matrice*,[*verbo d'azione*]

Commenti

verbo d'azione può essere PSET, PRESET, AND, OR o XOR.

(*x,y*) sono le coordinate dell'angolo superiore sinistro dell'immagine da trasferire.

Le istruzioni PUT e GET inviano e ricevono immagini grafiche dallo schermo. PUT e GET rendono possibile l'animazione ed il movimento di oggetti ad alta velocità nelle diverse modalità grafiche.

L'istruzione PUT trasferisce sullo schermo l'immagine memorizzata nella matrice. Il punto specificato è la coordinata dell'angolo superiore sinistro dell'immagine. Se l'ampiezza dell'immagine da trasferire supera le capacità dello schermo, si verifica l'errore `Chiamata di funzione illegale`.

Il *verbo d'azione* viene usato per l'interazione tra l'immagine trasferita e l'immagine già sullo schermo. PSET trasferisce i dati sullo schermo tali e quali.

PRESET è simile a PSET; l'unica differenza è che PRESET produce un immagine in "reverse" (nero su bianco).

AND trasferisce l'immagine soltanto se esiste già un'immagine sotto l'immagine trasferita.

OR sovrappone l'immagine all'immagine esistente.

XOR è una modalità speciale, spesso usata per l'animazione. XOR determina l'inversione dei punti sullo schermo se esiste un punto nell'immagine della matrice. Questo comportamento è esattamente analogo a quello del cursore sullo schermo.

XOR è specialmente utile per l'animazione. Quando un'immagine viene posta per due volte di seguito su un fondo complesso, il fondo viene ripristinato senza subire alcuna modifica. Un oggetto può perciò muoversi sullo schermo senza modificarne il fondo.

La modalità d'azione predefinita è XOR.

Per informazioni riguardanti gli effetti nelle diverse modalità, vedere le istruzioni COLOR, PALETTE e SCREEN.

L'animazione di un oggetto viene generalmente eseguita nel seguente modo:

1. Introdurre l'oggetto (o gli oggetti) sullo schermo.
2. Ricalcolare la nuova posizione dell'oggetto.
3. Reintrodurre l'oggetto sullo schermo alla posizione precedente, per rimuovere la vecchia immagine.
4. Ritornare all'operazione 1, inserendo questa volta l'oggetto nella nuova posizione.

Il movimento eseguito in questo modo lascia il fondo immutato. Il "farfallìo" può essere ridotto minimizzando il tempo tra l'operazione 4 e 1, ed assicurandosi che vi sia abbastanza tempo di pausa tra le operazioni 1 e 3. Se l'animazione riguarda più oggetti, eseguirli tutti insieme.

Se non è importante conservare il fondo, l'animazione può essere eseguita usando PSET.

Impostare intorno all'immagine un bordo che sia largo almeno quanto la distanza coperta dallo spostamento dell'oggetto. In questo modo, quando l'oggetto viene spostato, il bordo cancella qualsiasi punto. Questo metodo può essere più rapido di quello precedentemente descritto (XOR), in quanto lo spostamento di un oggetto richiede un solo PUT. Tuttavia, quest'ultimo metodo può essere utilizzato solo se l'immagine da scrivere è più grande dell'immagine già esistente.

Esempi

```
10 CLS:SCREEN 1
20 PSET (130,120)
30 DRAW "U25;E7;R20;D32;L6;U12;L14"
40 DRAW "D12;L6":PSET(137,102)
50 DRAW "U4;E4;R8;D8;L12"
60 PSET (137,88)
70 DRAW "E4;R20;D32;G4":PAINT (131,119)
80 DIM A (500)
90 GET (125,130)-(170,80),A
100 FOR I= 1 TO 1000:NEXT I
110 PUT (20,20),A,PSET
120 FOR I= 1 TO 1000:NEXT i
130 GET (125,130)-(170,80),A
140 FOR I= 1 TO 1000:NEXT I
150 PUT (220,130),A,PRESET
```

RANDOMIZE (istruzione)

Funzione

Imposta il generatore di numeri casuali.

Sintassi

RANDOMIZE [*espressione*]
RANDOMIZE TIMER

Commenti

Se *espressione* viene omessa, GW-BASIC sospende l'esecuzione e chiede il valore visualizzando la riga seguente:

```
Origine numero casuale (da -32768 a 32767)?
```

Se il generatore dei numeri casuali non viene reimpostato, ogni volta che il programma viene eseguito, la funzione RND restituisce la stessa sequenza di numeri casuali.

Per cambiare opportunamente la sequenza dei numeri casuali, porre l'istruzione RANDOMIZE all'inizio del programma e cambiare l'argomento ogni volta che lo si esegue (si veda la funzione RND).

RANDOMIZE privo di argomenti solleciterà l'inserimento del valore. RANDOMIZE [*espressione*] non trasformerà i valori a punto mobile in valori interi. *espressione* può essere una qualsiasi formula numerica.

Per ottenere la nuova origine dei numeri casuali senza il sollecito sopra menzionato, usare la funzione numerica TIMER nel seguente modo:

```
RANDOMIZE TIMER
```

Esempio 1

L'orologio interno può essere impostato ad intervalli.

```
10 RANDOMIZE TIMER
20 FOR I=1 TO 5
30 PRINT RND;
40 NEXT I
RUN
.88598       .484668  .586328  .119426  .709225
Ok
RUN
.803506      .162462  .929364  .292443  .322921
Ok
```

Esempio 2

L'orologio interno può essere usato come valore iniziale.

```
5  N=VAL (MID$(TIME$,7,2))    'utilizza i secondi come
   valore iniziale
10 RANDOMIZE N                'imposta il numero
20 PRINT RND                  'stampa i secondi
30 PRINT RND                  'stampa i numeri casuali
RUN
36
.2466638
Ok
RUN
37
.6530511
Ok
RUN
38
5.943847+02
Ok
RUN
40
.8722131
Ok
```

READ (istruzione)

Funzione

Legge dei valori da un'istruzione DATA e li assegna alle variabili.

Sintassi

READ *elencovariabili*

Commenti

L'istruzione READ deve essere usata sempre in combinazione con l'istruzione DATA.

L'istruzione READ assegna una variabile per ogni valore dell'istruzione DATA.

Le variabili per l'istruzione READ possono essere numeriche o alfanumeriche, ed i valori letti devono essere coerenti con i tipi di variabili specificati. Altrimenti, si verifica un `Errore di sintassi.`

Una sola istruzione READ può accedere a più istruzioni DATA. L'accesso avviene in ordine di specificazione. Varie istruzioni READ possono accedere ad una singola istruzione DATA.

Se il numero delle variabili in *elencovariabili* è più alto del numero degli elementi specificati nell'istruzione (o nelle istruzioni) DATA, le istruzioni READ seguenti iniziano la lettura dei dati dal primo elemento non letto. Se non vi sono ulteriori istruzioni READ, i dati in eccedenza vengono ignorati.

Per rileggere le istruzioni DATA dall'inizio, usare l'istruzione RESTORE.

Esempi

```
.
.
.
80 FOR I=1 TO 10
90 READ A(I)
100 NEXT I
110 DATA 3.08,5.19,3.12,3.98,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
.
.
.
```

Questa porzione di programma legge i valori dalle istruzioni DATA nella matrice A. Dopo l'esecuzione, il valore di A(1) è 3.08, e così per le altre variabili. Le istruzioni DATA (righe 110 e 120) possono essere poste in qualsiasi punto del programma, anche prima dell'istruzione READ.

```
5 PRINT
10 PRINT "CITTA'","REGIONE","CAP"
20 READ C$,R$,P
30 DATA "BERGAMO,","LOMBARDIA",24100
40 PRINT C$,R$,P
RUN

CITTA'   REGIONE  CAP
BERGAMO,LOMBARDIO  80211
Ok
```

Questo programma legge dati alfanumerici e numerici dall'istruzione DATA alla riga 30.

REM (istruzione)

Funzione

Permette l'inserimento di commenti e spiegazioni nel programma.

Sintassi

REM[*commento*]
'[*commento*]

Commenti

Le istruzioni REM non vengono eseguite, ma appaiono esattamente come inserite quando il programma viene visualizzato.

Una volta incontrata l'istruzione REM o la sua abbreviazione, l'apostrofo ('), il programma ignora tutto fino alla riga eseguibile seguente, o fino all'istruzione END.

Ci si può ricollegare ad istruzioni REM mediante istruzioni GOTO o GOSUB. Dopo il collegamento, l'esecuzione del programma continua con la prima istruzione eseguibile successiva a REM. Tuttavia, se ci si ricollega direttamente ad un'istruzione eseguibile, l'esecuzione del programma risulta più rapida.

I commenti possono essere aggiunti alla fine della riga, inserendo un apostrofo (') invece di REM.

Avvertenza Non usare REM in un'istruzione DATA, in quanto il contenuto verrebbe considerato un dato.

Esempi

```
.
.
.
120 REM CALCOLO DELLA VELOCITA' MEDIA
130 FOR I=1 TO 20
440 SUM=SUM+V(I)
450 NEXT I
.
.
.
```

oppure

```
.
.
.
129 FOR I=1 TO 20 'MEDIA CALCOLATA
130 SUM=SUM+V(I)
140 NEXT I
.
.
.
```

RENUM (comando)

Funzione

Rinumera le righe del programma.

Sintassi

RENUM[*numeronuovo*],[*numerovecchio*][,*incremento***R**]]

Commenti

numeronuovo è il primo numero di riga da usare nella nuova sequenza. Il valore predefinito è 10.

numerovecchio è la riga nel programma corrente da dove deve avere inizio la rinumerazione. Il valore predefinito è la prima riga del programma.

incremento è l'incremento da usare nella nuova sequenza. Il valore predefinito è 10.

Il comando RENUM cambia anche tutti i riferimenti a numeri di riga contenuti in istruzioni ELSE, GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB, RESTORE, RESUME ed ERL, in accordo alla nuova numerazione. Se dopo una delle istruzioni elencate appare un numero di riga non esistente, viene visualizzato il messaggio d'errore `Riga non definita x alla riga` *y*. Il numero di riga errato *x* non viene sostituito da RENUM, ma può essere cambiato il numero di riga *y*.

RENUM non può essere usato per cambiare l'ordine delle righe del programma (ad esempio, *RENUM 15,30* quando il programma ha tre righe numerate 10, 20 e 30) o per creare numeri di riga superiori a 65529. Si determinerebbe in questo caso una `Chiamata di funzione illegale.`

Esempi

```
RENUM
```

Rinumera l'intero programma. Il primo numero di riga sarà 10. Le righe successive vengono incrementate di 10.

```
RENUM 300,,50
```

Rinumera l'intero programma. Il primo numero di riga sarà 300. L'incremento è di 50 ogni riga.

```
RENUM 1000,900,20
```

Rinumera le righe da 900 in poi, iniziando dal numero di riga 1000 e con un incremento di 20.

RESET (comando)

Funzione

Chiude tutti i file di disco e scrive le informazioni della directory sul dischetto prima che esso venga rimosso dall'unità disco.

Sintassi

RESET

Commenti

Prima di rimuovere un dischetto dall'unità, eseguire sempre il comando RESET. Altrimenti, quando il dischetto verrà riutilizzato, non conterrà più le informazioni correnti relative alla directory sulla pista della directory.

RESET chiude tutti i file in tutte le unità disco e scrive la pista della directory su ogni dischetto con file aperti.

RESTORE (istruzione)

Permette la rilettura delle istruzioni DATA a partire dalla riga specificata.

Sintassi

RESTORE[*numeroriga*]

Commenti

Se *numeroriga* viene specificato, l'istruzione READ seguente ha accesso al primo elemento nell'istruzione DATA specificata.

Se *numeroriga* viene omesso, l'istruzione READ che segue accede al primo elemento nella prima istruzione DATA.

Esempi

```
10 READ A,B,C,
20 RESTORE
30 READ D,E,F
40 DATA 57,68,79
.
.
.
```

Questo segmento di programma assegna il valore 57 ad entrambe le variabili A e D, il valore 68 a B ed E, e così via.

RESUME (istruzione)

Funzione

Continua l'esecuzione del programma dopo che una routine per la correzione di errori è stata eseguita.

Sintassi

RESUME
RESUME 0
RESUME NEXT
RESUME ***numeroriga***

Commenti

Il formato dipende dalla posizione dalla quale l'esecuzione deve riprendere:

Sintassi	*Risultato*
RESUME o RESUME 0	L'esecuzione riprende dall'istruzione che ha causato l'errore.
RESUME NEXT	L'esecuzione riprende dall'istruzione immediatamente successiva all'istruzione che ha causato l'errore.
RESUME *numeroriga*	L'esecuzione riprende dal numero di riga specificato.

Un'istruzione RESUME non inclusa in una routine per il trapping di errori determina la visualizzazione del messaggio `RESUME senza errore`.

Esempi

```
10 ON ERROR GOTO 900
.
.
.
900 IF (ERR=230)AND(ERL=90) THEN PRINT
"RITENTA":RESUME 80
.
.
.
```

Se si verifica un errore dopo l'esecuzione della riga 10, viene eseguita l'azione indicata alla riga 900 ed il programma continua dalla riga 80.

RETURN (istruzione)

Funzione

Ritorna da una subroutine.

Sintassi

RETURN [*numeroriga*]

Commenti

L'istruzione RETURN determina il collegamento di GW-BASIC all'istruzione che segue l'istruzione GOSUB più recente. Una subroutine può contenere più di un'istruzione RETURN, al fine di permettere di ritornare da punti differenti della subroutine. Le subroutine possono essere inserite in qualsiasi parte del programma.

L'opzione *numeroriga* viene principalmente usata per il trapping di eventi. Questa istruzione rimanda la routine per il trapping di eventi ad un numero di riga specificato nel programma di GW-BASIC pur eliminando la voce di GOSUB creata dalla routine.

Quando viene creato un trapping per un evento particolare, la routine determina uno STOP automatico su quell'evento, in modo da evitare trapping ricorsivi. L'istruzione RETURN esegue un ON automaticamente a meno che non venga esplicitamente specificato un OFF all'interno della routine di trapping.

Il RETURN privo di *numeroriga* deve essere usato con attenzione. Qualsiasi istruzione GOSUB, WHILE o FOR attiva durante il trapping rimane attiva.

RIGHT$ (funzione)

Funzione

Restituisce gli *i* caratteri più a destra nella stringa x$.

Sintassi

RIGHT$(x$,*i*)

Commenti

Se *i* è uguale o maggiore di LEN(x$), RIGHT$ restituisce x$. Se *i* è uguale a zero, viene restituita la stringa nulla (di lunghezza zero). Si vedano le istruzioni MID$ e LEFT$.

Esempi

```
10 A$="DISCO BASIC"
20 PRINT RIGHT$(A$,5)
RUN
BASIC
Ok
```

Questo stampa i cinque caratteri più a destra della stringa A$.

RMDIR (comando)

Funzione

Cancella una subdirectory.

Sintassi

RMDIR *nomepercorso*

Commenti

nomepercorso è un'espressione alfanumerica, lunga non più di 63 caratteri, che identifica la subdirectory da rimuovere dalla directory madre.

La subdirectory da cancellare non deve contenere alcun file diverso da "." e "..". In caso contrario si verifica l'errore `Errore di accesso al percorso/file`.

Esempio

Il comando seguente cancella la subdirectory CLIENTI:

```
RMDIR "FATTURE\CLIENTI"
```

RND (funzione)

Funzione

Restituisce un numero casuale tra 0 e 1.

Sintassi

RND[(*x*)]

Commenti

Ad ogni esecuzione di un programma viene generata la stessa sequenza di numeri casuali, a meno che il generatore di numeri casuali non venga reimpostato (si veda l'istruzione RANDOMIZE). Se *x* è uguale a 0, l'ultimo numero viene ripetuto.

Se *x* è maggiore di 0 o se il valore *x* viene omesso, viene generato il numero casuale successivo.

Per ottenere un numero casuale nell'intervallo da 0 a *n*, usare la formula seguente:

INT(RND*(n+1))

Il generatore di numeri casuali può essere impostato usando un valore di *x* negativo.

Esempi

```
10 FOR I=1 TO 5
20 PRINT INT(RND*101);
30 NEXT
RUN
53     30     31     51     5
Ok
```

Genera cinque numeri pseudo-casuali nell'intervallo tra 0 e 100.

RUN (comando)

Funzione

Esegue il programma correntemente in memoria o carica in memoria un file dal dischetto e lo esegue.

Sintassi

RUN [*numeroriga*][**,r**]
RUN *nomefile*[**,r**]

Commenti

RUN e RUN *numeroriga* eseguono il programma correntemente in memoria.

Se viene specificato *numeroriga*, l'esecuzione inizia da quella riga. In caso contrario, l'esecuzione inizia dal numero di riga più basso.

Se in memoria non vi è alcun programma quando viene eseguito RUN, GW-BASIC ritorna al livello di comando.

RUN *nomefile* chiude tutti i file aperti e, prima di caricare il file in memoria specificato ed eseguirlo, cancella il contenuto corrente della memoria.

L'opzione **r** mantiene tutti i file di dati aperti.

Se si usa una cassa acustica, l'esecuzione del comando RUN determina la disattivazione di qualsiasi suono correntemente in esecuzione, seguita dalla reimpostazione in musica di primo piano. Inoltre, le istruzioni PEN e STRIG vengono disattivate.

Esempi

```
RUN NUOVOFIL,R
```

Esegue NUOVOFIL senza chiudere i file di dati.

SAVE (comando)

Funzione

Salva un file programma sul dischetto.

Sintassi

SAVE *nomefile*[**,a**]
SAVE *nomefile*[**,p**]

Commenti

nomefile è una stringa racchiusa tra virgolette che segue le normali regole di MS-DOS per l'assegnazione di nomi di file. Se il nome specificato esiste già, il file preesistente viene sovrascritto. Se viene omessa l'estensione, viene usata .BAS.

L'opzione **a** salva il file nel formato ASCII. Se si omette questa opzione, GW-BASIC salva il file automaticamente nel formato binario compresso. Se si usa il formato ASCII, si occupa più spazio sul dischetto; tuttavia, alcuni comandi per l'accesso ai dischetti (ad esempio, il comando MERGE ed alcuni comandi MS-DOS, del tipo TYPE) potrebbero richiederlo.

L'opzione **p** protegge il file salvandolo in un formato binario codificato. Se un file protetto viene eseguito o caricato, qualsiasi tentativo di modifica o di visualizzazione verrà respinto. Quando si usa l'opzione **p**, creare una copia del file sotto un altro nome o su un dischetto diverso, in modo da facilitare la manutenzione del programma.

Esempi

Il comando seguente salva il file COM2.BAS nel formato ASCII:

```
SAVE COM2,A
```

Il comando seguente salva il file PROG.BAS in formato binario e ne protegge l'accesso.

```
SAVE PROG,P
```

SCREEN (funzione)

Funzione

Restituisce il codice ASCII (da 0 a 255) del carattere che si trova nella riga e nella colonna specificata.

Sintassi

x=SCREEN(*riga*,*colonna*[,*z*])

Commenti

x è una variabile numerica che riceve il codice ASCII restituito.

riga è un'espressione numerica valida nell'intervallo da 1 a 25.

col è un'espressione numerica valida nell'intervallo da 1 a 40 o da 1 a 80, secondo l'impostazione della larghezza dello schermo. Si veda l'istruzione WIDTH per ulteriori dettagli.

z è un'espressione numerica valida contenente un valore vero o falso. Questa espressione può essere utilizzata soltanto in modalità Alfa.

L'ordinale del carattere che si trova alle coordinate specificate viene memorizzato nella variabile numerica. Se viene fornito il parametro opzionale *z* vero (non zero) e si è in modalità Alfa, invece del codice ASCII, viene restituito l'attributo di colore per il carattere (si veda l'istruzione COLOR).

L'inserimento di qualsiasi valore che non rientra nell'intervallo specificato causa una `Chiamata di funzione illegale`. Si può inserire un riferimento alla riga 25 soltanto se il tasto di funzione è disattivato.

Esempi

```
100 X=SCREEN (10,10)
```

Se il carattere che si trova in (10,10) è A, X vale 65.

```
110 X= SCREEN (1,1,1)
```

Restituisce l'attributo di colore del carattere che si trova nell'angolo superiore sinistro dello schermo.

SCREEN (istruzione)

Funzione

Imposta le specificazioni per la visualizzazione.

Sintassi

SCREEN [*modalità*][,[*colore*]][,[*apagina*]][,[*vpagina*]]

Commenti

L'istruzione SCREEN viene soprattutto usata per la selezione della modalità di schermo appropriata per una determinata configurazione di hardware. Le configurazioni di hardware e le modalità di schermo supportate sono descritte sotto.

MDPA con monitor monocolore: modalità 0

Il Monocolor Display and Printer Adapter (MDPA) IBM viene usato solo per visualizzazione monocolore. I programmi scritti per questa configurazione devono essere unicamente in modalità di testo.

CGA con Color Display: modalità 0, 1 e 2

Il Color Graphics Adapter (CGA) IBM e il Color Display vengono normalmente abbinati. Questa configurazione di hardware permette l'esecuzione dei programmi in modalità di testo e quella di programmi grafici a risoluzione media e ad alta risoluzione.

EGA con Color Display: modalità 0, 1, 2, 7 e 8

Quando l'Enhanced Graphics Adapter (EGA) IBM viene collegato al Color Display, le cinque modalità di schermo, 0, 1, 2, 7 ed 8, permettono l'interfaccia con il monitor. Se i parametri EGA vengono impostati per la compatibilità con CGA, i programmi scritti per le modalità 1 e 2 vengono eseguiti allo stesso modo in cui verrebbero eseguiti con il CGA. Le modalità 7 e 8 sono analoghe alle modalità 1 e 2; l'unica differenza è che con le modalità 7 e 8 si può disporre di una gamma di colori più vasta.

EGA con Enhanced Color Display: modalità 0, 1, 2, 7 e 8

Con la configurazione Enhanced Color Display/EGA IBM, le modalità 0, 1, 2, 7 e 8 sono virtualmente identiche a quelle del sistema Color Display/EGA. Tuttavia, vi sono due differenze possibili:

1. In modalità 0, il colore del bordo può non essere lo stesso del Color Display/EGA, in quanto l'impostazione del bordo su un Enhanced Color Display in modalità di testo 640 X 350 non è possibile.
2. La qualità del testo è migliore su un Enhanced Color Display. Infatti quest'ultimo dispone di una matrice di carattere 8 X 14, mentre la matrice di un Color Display è 8 X 8.

EGA con Enhanced Color Display: modalità 9

In questa modalità, la capacità dell'Enhanced Color Display viene sfruttata interamente. La modalità 9 permette la risoluzione più alta possibile per la configurazione Enhanced Color Display/EGA. I programmi scritti per questa modalità non possono essere utilizzati con nessun'altra configurazione di hardware.

EGA con monitor monocolore: modalità 10

In questa modalità il monitor monocolore IBM può essere usato per visualizzare grafici monocolori a risoluzione molto alta. I programmi scritti per questa modalità non possono essere usati con nessun'altra configurazione di hardware.

Argomenti

L'argomento *modalità* è un valore intero da scegliersi tra i seguenti: 0, 1, 2, 7, 8, 9 e 10. Tutti gli altri valori non sono leciti. La selezione di una modalità dipende principalmente dall'hardware di cui si dispone, come descritto precedentemente.

I paragrafi che seguono contengono una descrizione di ciascuna modalità di schermo.

SCREEN 0

- Soltanto modalità di testo
- Formato del testo 40 X 25 o 80 X 25 con matrice di carattere 8 X 8 (8 X 14 con EGA)
- Assegnazione di 16 colori a due attributi
- Assegnazione di 16 colori a 16 attributi (con EGA)

SCREEN 1

- Grafici a risoluzione media 320 X 200 pixel
- Formato di testo 80 X 25 con matrice di carattere 8 X 8
- Assegnazione di 16 colori a 4 attributi
- Supporta EGA e CGA
- 2 bit per pixel

SCREEN 2

- Grafici ad alta risoluzione 640 X 200 pixel
- Formato di testo 40 X 25 e matrice di carattere 8 X 8
- Assegnazione di 16 colori a 2 attributi
- Supporta EGA e CGA
- 1 bit per pixel

SCREEN 7

- Grafici a risoluzione media 320 X 200 pixel
- Formato di testo 40 X 25 e matrice di carattere 8 X 8
- 2, 4 o 8 pagine di memoria con rispettivamente 64K, 128K o 256K di memoria, installati sull'EGA
- Assegnazione di tutti i 16 colori a 16 attributi
- E' necessario l'utilizzo dell'EGA
- 4 bit per pixel

SCREEN 8

- Grafici ad alta risoluzione 640 X 200 pixel
- Formato di testo 80 X 25 e matrice di carattere 8 X 8
- 1, 2 o 4 pagine di memoria con rispettivamente 64K, 128K, o 256K di memoria, installati sull'EGA
- Assegnazione di tutti i 16 colori a 16 attributi
- E' necessario l'utilizzo dell'EGA
- 4 bit per pixel

SCREEN 9

- Grafici a risoluzione avanzata 640 X 350 pixel
- Formato di testo 80 X 25 e matrice di carattere 8 X 14
- Assegnazione di 64 colori a 16 attributi (oltre 64K di memoria di EGA), o 16 colori a 4 attributi (64K di memoria di EGA)
- Con 256K di memoria di EGA installati, due pagine di visualizzazione
- E' necessario l'utilizzo dell'EGA
- 2 bit per pixel (64K di memoria di EGA)
- 4 bit per pixel (oltre 64K di memoria di EGA)

SCREEN 10

- Grafici a risoluzione avanzata 640 X 350 pixel
- Formato di testo 80 X 25 e matrice di carattere 8 X 14
- Con 256K di EGA di memoria installati, due pagine di visualizzazione
- Assegnazione di un massimo di 9 pseudo-colori a 4 attributi
- E' necessario l'utilizzo dell'EGA
- 2 bit per pixel

I seguenti sono gli attributi predefiniti per SCREEN 10, monitor monocolore:

Valore dell'attributo	*Pseudo-colore visualizzato*
0	Spento
1	Acceso, intensità normale
2	Intermittente
3	Acceso, alta intensità

I seguenti sono i valori per i colori per SCREEN 10, monitor monocolore:

Valore del colore	*Pseudo-colore visualizzato*
0	Spento
1	Intermittente, spento/acceso
2	Intermittente, spento/alta intensità
3	Intermittente, acceso/spento
4	Acceso
5	Intermittente, acceso/alta intensità
6	Intermittente, alta intensità/spento
7	Intermittente, alta intensità/acceso
8	Alta intensità

Sia per i monitor compositi che per le televisioni, il *colore* è un'espressione numerica vera (non zero) o falsa (zero). Il valore zero disattiva il colore e permette unicamente la visualizzazione di immagini in bianco e nero. Un valore non zero, invece, permette le visualizzazioni a colori. Nella modalità SCREEN 0, il significato del parametro *colore* viene invertito.

Per le configurazioni di hardware che hanno un EGA ed abbastanza memoria da supportare pagine di schermo molteplici, sono disponibili due altri parametri: *apagina* e *vpagina*. Essi determinano le pagine di memoria "attive" e "visuali". La pagina attiva è la parte di memoria dove vengono scritte istruzioni per grafici; la pagina visuale è la parte di memoria che viene visualizzata sullo schermo.

Alternando la visualizzazione delle pagine grafiche è possibile ottenere l'animazione. L'obbiettivo è di visualizzare la pagina visuale con un output grafico completo, durante l'esecuzione delle istruzioni grafiche in una o più pagine. La visualizzazione di una pagina viene eseguita soltanto quando l'output grafico di quella pagina è terminato. Per cui, è tipico il segmento di programma seguente:

```
SCREEN 7,,1,2'esegui pag 1, visualizza pag 2
.
.  L'output grafico a pagina 1
.  durante la visione di pagina 2
```

```
SCREEN 7,,2,1'esegui pagina 2, visualizza pagina 1
.
.  L'output dei grafici in pagina 2
.  durante la visione di pagina 1
.
```

Il numero delle pagine disponibili dipende dalla modalità SCREEN e dalla quantità di memoria disponibile, come spiegato nella tabella seguente:

Tabella 2 Specificazioni delle modalità di schermo

Modalità	*Risoluzione*	*Intervallo attributi*	*Intervallo colori*	*Memoria EGA*	*Pagine*	*Dimensione pagine*
0	testo a 40 colonne	NA	da 0 a 15[a]	NA	1	2K
	testo ad 80 colonne	NA	da 0 a 15[a]	NA	1	4K
1	320 X 200	da 0 a 3[b]	da 0 a 3	NA	1	16K
2	640 X 200	da 0 a 1[b]	da 0 a 1	NA	1	16K
7	320 X 200	da 0 a 15	da 0 a 15	64K	2	32K
				128K	4	
				256K	8	
8	640 X 200	da 0 a 15	da 0 a 15	64K	1	64K
				128K	2	
				256K	4	
9	640 X 350	da 0 a 3	da 0 a 15	64K	1	64K
		da 0 a 15	da 0 a 63	128K	1	128K
		da 0 a 15	da 0 a 63	256K	2	
10	640 X 350	da 0 a 3	da 0 a 8	128K	1	128K
				256K	2	

a I numeri da 16 a 31 sono la versione intermittente dei colori da 0 a 15.

b Attributi applicabili soltanto con EGA.

Attributi e colori

Per le varie modalità di schermo e configurazioni di hardware, vi sono diverse impostazioni di attributi e colori. Per una dettagliata spiegazione riguardante attributi e numeri di colori, si veda l'istruzione PALETTE. La maggior parte di questi attributi e configurazioni sono raccolte nella tabella seguente:

Tabella 3 Attributi e colori predefiniti per la maggior parte delle modalità di schermo

Attributi di modalità			***Monitor a colori***		***Monitor monocolore***	
1,9	***2***	***0,7,8,9[b]***	***Numero[c]***	***Colore***	***Numero[c]***	***Colore***
0	0	0	0	Nero	0	Spento
		1	1	Blu		(Sottolineato)[a]
		2	2	Verde	1	Acceso[a]
		3	3	Turchese	1	Acceso[a]
		4	4	Rosso	1	Acceso[a]
		5	5	Magenta	1	Acceso[a]
		6	6	Marrone	1	Acceso[a]
		7	7	Bianco	1	Acceso[a]
		8	8	Grigio	0	Spento
		9	9	Azzurro chiaro		Intensità alta (sottolineato)
		10	10	Verde chiaro	2	Intensità alta
1		11	11	Turchese chiaro	2	Intensità alta
		12	12	Rosso chiaro	2	Intensità alta
2		13	13	Magenta chiaro	2	Intensità alta
		14	14	Giallo	2	Intensità alta
3	1	15	15	Intensità alta Bianco	0	Spento

a Spento quando usato come fondo.

b Con memoria di EGA superiore a 64K

c Soltanto per la modalità 0 monocolore.

I colori di primo piano predefiniti per le varie modalità sono raccolti nella tabella seguente:

Tabella 4 Colori di primo piano predefiniti

Attributo di primo piano predefinito			*Colore di primo piano predefinito*	
Modalità schermo	*Color/ Enhanced[a] Display*	*Monitor monocolore*	*Color/ Enhanced[a] Display*	*Monitor monocolore*
0	7	7	7	1
1	3	NA	15	NA
2	1	NA	15	NA
7	15	NA	15	NA
8	15	NA	15	NA
9	3[b]	NA	63	NA
10	NA	3	NA	8

a Enhanced Color Display IBM

b 15 se vi è più di 64K di memoria di EGA

NA=Non Applicabile

SGN (funzione)

Funzione

Restituisce il segno di *x*.

Sintassi

SGN(*x*)

Commenti

x è una qualsiasi espressione numerica.

Se *x* è positivo, SGN(x) restituisce 1.
Se *x* è 0, SGN(x) restituisce 0.
Se *x* è negativo, SGN(x) restituisce -1.

Esempi

```
10 INPUT "Inserire il valore",x
20 ON SGN(X)+2 GOTO 100,200,300
```

Se X è negativo, GW-BASIC si ricollega alla riga 100; se X è uguale a 0, il collegamento avviene alla riga 200; se infine X è positivo GW-BASIC si sposta alla riga 300.

SHELL (istruzione)

Funzione

Carica ed esegue un altro programma o un file batch. Al termine del programma, il controllo viene restituito al programma di GW-BASIC, o più esattamente alla istruzione che segue SHELL. Un programma eseguito sotto il controllo di GW-BASIC viene normalmente definito come **esecuzione di figlia**.

Sintassi

SHELL[*stringa*]

Commenti

stringa è un'espressione alfanumerica valida contenente il nome del programma da eseguire e eventualmente gli argomenti di comando.

Il nome del programma in *stringa* può avere qualsiasi estensione supportata dal processore di comando di MS-DOS. Se l'estensione non viene specificata, COMMAND.COM cerca, nell'ordine, un file .COM, .EXE o .BAT. Se nessuno di questi file viene trovato, l'istruzione SHELL visualizza il messaggio d'errore `File non trovato.`

Qualsiasi testo separato dal nome del programma da almeno uno spazio, viene interpretato da COMMAND.COM come parametro del programma.

Durante l'esecuzione della figlia, GW-BASIC rimane in memoria. Non appena l'esecuzione giunge al termine, GW-BASIC riprende dall'istruzione che segue SHELL.

SHELL senza stringhe accede a MS-DOS e rende possibile l'esecuzione di qualsiasi operazione permessa da COMMAND. Quando si è pronti per ritornare a GW-BASIC, digitare il comando MS-DOS Exit.

Esempi

```
SHELL
A>DIR
A>EXIT
Ok
```

Scrivere ad esempio dei dati da ordinare, usare SHELL SORT per eseguirne l'ordinamento e leggere poi i dati ordinati per scrivere una relazione.

```
10 OPEN "SORTIN.DAT" FOR OUTPUT AS #1
20 'Scrivere i dati da ordinare
.
.
.
1000 CLOSE 1
1010 SHELL "SORT <SORTIN.DAT  >SORTOUT.DAT"
1020 OPEN "SORTOUT.DAT" FOR INPUT AS #1
1030  'Procedere sui dati ordinati
```

SIN (funzione)

Funzione

Calcola il seno trigonometrico di *x*, in radianti.

Sintassi

SIN(*x*)

Commenti

SIN(*x*) viene calcolato in precisione singola a meno che non sia stato usato il parametro **/d** all'esecuzione di GW-BASIC.

Per ottenere SIN(*x*) quando *x* è espressa in gradi, usare SIN(*x**π/180).

Esempi

```
PRINT SIN(1.5)
.9974951
Ok
```

Il seno di 1.5 radianti è 0.9974951 (precisione singola).

SOUND (istruzione)

Funzione

Genera il suono attraverso la cassa acustica.

Sintassi

SOUND *freq,durata*

Commenti

freq è la frequenza desiderata in Hertz (cicli per secondo). Si tratta di un'espressione numerica nell'intervallo da 37 a 32767.

durata è la durata desiderata in impulsi di orologio. Gli impulsi di orologio vengono ripetuti 18.2 volte al secondo. *durata* deve essere un'espressione numerica compresa tra 0 e 65535.

I valori al di sotto di 0.022 producono il suono fino all'esecuzione di una nuova istruzione SOUND o PLAY.

Se *durata* è zero, qualsiasi istruzione SOUND attiva viene disattivata. Se però l'istruzione SOUND è in fase di esecuzione, la *durata* zero non ha alcun effetto.

Il suono viene eseguito in primo piano o in sottofondo in relazione all'istruzione PLAY.

Esempi

L'esempio seguente crea suoni casuali di breve durata:

```
2500 SOUND RND*1000+37,2
2600 GOTO 2500
```

La tabella seguente mostra il rapporto esistente tra le note e le loro frequenze nei due ottavi adiacenti a Do3.

Tabella 5 Il rapporto tra le note e le frequenze

Nota	*Frequenza*	*Nota*	*Frequenza*
C	130.810	C*	523.250
D	146.830	D	587.330
E	164.810	E	659.260
F	174.610	F	698.460
G	196.000	G	783.990
A	220.000	A	880.000
B	246.940	B	987.770
C	261.630	C	1046.500
D	293.660	D	1174.700
E	329.630	E	1318.500
F	349.230	F	1396.900
G	392.000	G	1568.000
A	440.000	A	1760.000
B	493.880	B	1975.500

*Do3.

Raddoppiando o dimezzando la frequenza, i valori delle note coincidenti possono essere stimati per le ottave che precedono e che seguono.

Per creare dei periodi di silenzio, usare l'istruzione seguente:

`SOUND 32767,`*durata*

Per calcolare la durata di un battito, dividere i battiti per minuto per il numero degli impulsi (tic) di orologio in un minuto (1092).

La tabella seguente mostra i tempi richiesti dagli impulsi di orologio:

Tabella 6 Tempi richiesti dagli impulsi di orologio

Tempo	***Notazione***	***Battiti per minuto***	***Impulsi per battito***
Molto lento	Larghissimo		
	Largo	40-66	27.3-18.02
	Larghetto	60-66	18.2-18.55
	Grave		
	Lento		
	Adagio	66-76	16.55-14.37
Lento	Adagietto		
	Andante	76-108	14.37-10.11
Medio	Andantino		
	Moderato	108-120	10.11-9.1
Veloce	Allegretto		
	Allegro	120-168	9.1-6.5
	Vivace		
	Veloce		
	Presto	168-208	6.5-5.25
Molto veloce	Prestissimo		

SPACE$ (funzione)

Funzione

Restituisce una stringa di x spazi.

Sintassi

SPACE$(*x*)

Commenti

x viene arrotondato ad un valore intero e deve rientrare nell'intervallo da 0 a 255 (si veda la funzione SPC).

Esempi

```
10 FOR N=1 TO 5
20 X$=SPACE$(N)
30 PRINT X$;N
40 NEXT N
RUN
1
 2
  3
   4
    5
Ok
```

La riga 20 aggiunge uno spazio ad ogni esecuzione del ciclo.

SPC (funzione)

Funzione

Salta un numero di spazi specificato, nell'istruzione PRINT o LPRINT.

Sintassi

SPC(*n*)

Commenti

n deve rientrare nell'intervallo da 0 a 255.

Se *n* eccede la larghezza della stampante o dello schermo, il valore usato è la larghezza di *n* MOD.

Si assume che il comando SPC(*n*) sia seguito da un punto e virgola.

SPC può essere usato soltanto con le istruzioni PRINT, LPRINT e PRINT# (si veda la funzione SPACE$).

Esempi

```
PRINT "A" SPC(15) "DESTRA"
A             DESTRA
Ok
```

SQR (funzione)

Funzione

Calcola la radice quadrata di x.

Sintassi

SQR(x)

Commenti

x deve essere maggiore o uguale a 0.

SQR(x) viene calcolato in precisione singola a meno che all'esecuzione di GW-BASIC non sia stato usato il parametro **/d**.

Esempi

```
10 FOR X=10 TO 25 STEP 5
20 PRINT X; SQR(X)
30 NEXT
RUN
10          3.162278
15          3.872984
20          4.472136
25          5
Ok
```

STICK (funzione)

Funzione

Restituisce le coordinate di x e di y di due joystick.

Sintassi

x=STICK(*n*)

Commenti

x è una variabile numerica usata per memorizzare il risultato.

n è un'espressione numerica valida tra 0 e 3.

Valore di n	*Coordinata restituita*
0	Coordinata x del joystick A. Memorizza i valori di x e y di entrambi i joystick per le tre chiamate di funzioni seguenti.
1	Coordinata y del joystick A.
2	Coordinata x del joystick B
3	Coordinata y del joystick B.

STOP (istruzione)

Funzione

Termina l'esecuzione del programma e ritorna al livello di comando.

Sintassi

STOP

Commenti

Le istruzioni STOP possono essere usate in qualsiasi parte del programma per terminarne l'esecuzione. Quando il programma incontra STOP, viene stampato il messaggio seguente:

`Interruzione alla riga` *nnnnn*

A differenza dall'istruzione END, STOP non chiude i file.

Dopo l'esecuzione di STOP, GW-BASIC ritorna sempre al livello di comando. Eseguendo il comando CONT, l'esecuzione riprende.

Esempi

```
10 INPUT A,B,C
20 K=A^2*5.3:L=B^3/.26
30 STOP
40 M=C*K+100:PRINT M
RUN
? 1,2,3
INTERRUZIONE ALLA RIGA 30
Ok
PRINT L
30.76923
Ok
CONT
115.9
Ok
```

STR$ (funzione)

Funzione

Restituisce la rappresentazione alfanumerica del valore di x.

Sintassi

STR$(*x*)

Commenti

STR$(x) è la funzione complementare di VAL(x$). Si veda la funzione VAL.

Esempi

```
5 REM ARITMETICA PER BAMBINI
10 INPUT "DIGITARE UN NUMERO";N
20 ON LEN(STR$(N)) GOSUB 30,40,50
.
.
.
```

Questo programma si ricollega alle varie subroutine, in relazione al numero di caratteri digitati prima della pressione del tasto RITORNO.

STRIG (istruzione e funzione)

Funzione

Restituisce lo stato dei trigger del joystick.

Sintassi

Istruzione:

STRIG ON
STRIG OFF

Funzione:

x=**STRIG**(*n*)

Commenti

x è una variabile numerica per la memorizzazione del risultato.

n è un'espressione numerica tra 0 e 7.

Prima di effettuare qualsiasi chiamata della funzione STRIG(n), bisogna eseguire l'istruzione STRIG ON. Una volta eseguita STRIG ON, GW-BASIC controlla che sia stato premuto un tasto prima che qualsiasi altra istruzione venga eseguita. STRIG OFF disattiva il controllo.

n è un'espressione numerica tra 0 e 7, che restituisce i valori seguenti:

Valore di n	***Restituisce***
0	-1 se dopo l'ultima istruzione STRIG(0) è stato premuto il trigger A1; altrimenti restituisce 0.
1	-1 se il trigger A1 è premuto in quel momento; altrimenti restituisce 0.
2	-1 se dopo l'ultima istruzione STRIG(2) è stato premuto il trigger A1; altrimenti restituisce 0.
3	-1 se il trigger B1 è premuto in quel momento; altrimenti restituisce 0.
4	-1 se dopo l'ultima istruzione STRIG(4) è stato premuto il trigger A2; altrimenti restituisce 0.
5	-1 se il trigger A2 è premuto in quel momento; altrimenti, restituisce 0.

STRIG(n) (istruzione)

Funzione

Permette l'uso di un joystick attivandone o disattivandone il trapping dei tasti.

Sintassi

STRIG(*n*) ON
STRIG(*n*) OFF
STRIG(*n*) STOP

Commenti

n è 0, 2, 4 o 6, in relazione ai tasti del joystick:

0 è il tasto A1
2 è il tasto B1
4 è il tasto A2
6 è il tasto B2

Esempi

STRIG(*n*) ON

Attiva il trapping dei tasti del joystick. Dopo l'esecuzione di questa istruzione, GW-BASIC controlla se questo tasto è stato premuto, prima di eseguire le istruzioni successive.

STRIG(*n*) ON

Disattiva il controllo sopra descritto.

STRIG(*n*) STOP

Disattiva il trapping di un tasto impostato attraverso l'uso dell'istruzione ON STRIG(*n*). Tuttavia, l'eventuale pressione del tasto viene registrata e, non appena il controllo viene riattivato, si verifica il trapping.

STRING$ (funzione)

Funzione

Restituisce una stringa di lunghezza *n*, i cui caratteri hanno codice ASCII *j*, oppure una stringa della stessa lunghezza in cui tutti i caratteri corrispondono al primo carattere di x$.

Sintassi

STRING$(*n,j*)
**STRING$**(*n,x$*)

Commenti

La funzione STRING$ può anche essere utilizzata per stampare i bordi superiori e inferiori sulla stampante o sullo schermo.

n e *j* sono espressioni numeriche intere che rientrano nell'intervallo da 0 a 255.

Esempi

```
10 X$ = STRING$(10,45)
20 PRINT X$ "RELAZIONE MENSILE" X$
RUN
----------RELAZIONE MENSILE----------
Ok
```

45 è il valore decimale equivalente al simbolo ASCII meno (-).

Nell'appendice C del *Manuale dell'utente* sono elencati tutti i codici di carattere ASCII.

SWAP (istruzione)

Funzione

Scambia i valori di due variabili.

Sintassi

SWAP *variabile1,variabile2*

Commenti

E' possibile scambiare qualsiasi tipo di variabile (intera, a precisione singola, a precisione doppia e alfanumerica). Per eseguire un scambio, comunque, occorre che le due variabili siano dello stesso tipo. Altrimenti, si verifica l'errore `Tipi di carattere contrastanti.`

Esempi

```
LIST
10 A$="UNO ":B$="TUTTI ":C$="PER "
20 PRINT A$ C$ B$
30 SWAP A$, B$
40 PRINT A$ C$ B$
RUN
Ok
UNO PER TUTTI
TUTTI PER UNO
Ok
```

La riga 30 scambia i valori delle stringhe A$ e B$.

SYSTEM (comando)

Funzione

Permette il ritorno a MS-DOS.

Sintassi

SYSTEM

Commenti

Prima di premere RITORNO, **salvare il programma.** In caso contrario si perderà quanto elaborato nella corrente sessione di lavoro.

Prima di ritornare a MS-DOS, il comando SYSTEM chiude tutti i file. Se da MS-DOS si è entrati in GW-BASIC attraverso un file batch, il comando SYSTEM ritorna al file batch e ne continua l'esecuzione a partire dal punto di interruzione.

Esempi

```
SYSTEM
A>
```

TAB (funzione)

Funzione

Inserisce spazi fino al punto *n* dello schermo.

Sintassi

TAB(*n*)

Commenti

Se la posizione di stampa corrente è già oltre la posizione *n*, TAB si porta su tale posizione nella riga seguente.

Lo spazio 1 è la posizione più a sinistra. La posizione più a destra è il limite in larghezza dello schermo.

n deve rientrare nell'intervallo numerico da 1 a 255.

Se la funzione TAB si trova alla fine dell'elenco di dati, il cursore non viene trasferito alla riga seguente. TAB viene eseguito come se ci fosse un punto e virgola alla fine.

TAB può essere usato soltanto con le istruzioni PRINT, LPRINT, o PRINT# (si veda la funzione SPC).

Esempi

```
10 PRINT "NOME" TAB(25) "SOMMA": PRINT
20 READ A$,B$
30 PRINT A$ TAB(25) B$
40 DATA "G. P. ROSSI","27,500"
RUN
NOME                    SOMMA

G. P. ROSSI             27,500
Ok
```

TAN (funzione)

Funzione

Calcola la tangente trigonometrica di *x*, in radianti.

Sintassi

TAN(*x*)

Commenti

TAN(x) è calcolato in precisione singola a meno che all'esecuzione di GW-BASIC non sia stato usato il parametro **/d**.

Se il valore di TAN(x) eccede i limiti consentiti, viene visualizzato il messaggio `Valore troppo elevato`, viene fornito il valore infinito di macchina con il segno appropriato, e l'esecuzione continua.

Per ottenere TAN(*x*) quando *x* è espressa in gradi, usare TAN(*x**π/180).

Esempi

```
10 Y = TAN(X)
```

Dopo l'esecuzione, Y contiene il valore della tangente di X radianti.

TIME$ (variabile e istruzione)

Funzione

Visualizza e imposta l'orario corrente.

Sintassi

Istruzione:

TIME$=*espressione*

Variabile:

espressione=**TIME$**

Commenti

espressione è una stringa alfanumerica valida o una variabile, che permette l'impostazione di ore (*hh*), ore e minuti (*hh:mm*), oppure ore, minuti e secondi (*hh:mm:ss*).

hh imposta le ore (da 0 a 23). Il valore predefinito per minuti e secondi è 00.

hh:mm imposta le ore (da 0 a 23) ed i minuti (da 0 a 59). Il valore predefinito per i secondi è 00.

hh:mm:ss imposta le ore (da 0 a 23), i minuti (da 0 a 59) ed i secondi (da 0 a 59).

Se *espressione* non è una stringa valida, si verifica l'errore `Tipi di carattere contrastanti`.

Se nell'inserimento dei valori per l'ora vi sono zeri iniziali, questi possono essere eliminati. Per impostare ore, minuti o secondi al valore zero, bisogna comunque inserire almeno una cifra. Ad esempio, se si vuole impostare l'orario a mezzanotte e mezza, l'istruzione da inserire è: TIME$="0:30", e non TIME$= ":30".

Se uno dei valori eccede i limiti consentiti, si verifica l'errore `Chiamata di funzione illegale`. L'orario precedente non cambia.

Se TIME$ è la destinazione di una dichiarazione, viene memorizzata la data corrente.

Se TIME$ è l'espressione in un'istruzione LET o PRINT, l'orario corrente viene caricato ed assegnato alla variabile alfanumerica.

Con *espressione*=TIME$, TIME$ restituisce una stringa di 8 caratteri nel formato *hh:mm:ss*.

Esempi

L'esempio seguente imposta l'orario alle 8:00:

```
TIME$ = "08:00"
OK
PRINT TIME$
08:00:05
OK
```

Nel programma seguente vengono visualizzati la data e l'orario correnti sulla 25ma riga dello schermo; inoltre, verrà emesso un suono per ogni minuto e mezzo minuto trascorso.

```
10 KEY OFF:SCREEN 0:WIDTH 80:CLS
20 LOCATE 25,5
30 PRINT DATE$,TIME$;
40 SEC=VAL(MID$(ORA$,7,2))
50 IF SEC=SSEC THEN 20 ELSE SSEC=SEC
60 IF SEC=0 THEN 1010
70 IF SEC=30 THEN 1020
80 IF SEC<57 THEN 20

1000 SOUND 1000,2:GOTO 20
1010 SOUND 2000,8:GOTO 20
1020 SOUND 400,4:GOTO 20
```

TIMER (funzione)

Funzione

Restituisce numeri a precisione singola a punto mobile, che rappresentano i secondi trascorsi dalla mezzanotte o dalla reimpostazione del sistema.

Sintassi

v=TIMER

Commenti

Le frazioni di secondo vengono calcolate al grado di precisione più elevato possibile. TIMER è una funzione di sola lettura.

TRON/TROFF (comandi)

Funzione

Riproducono l'esecuzione delle istruzioni del programma.

Sintassi

TRON
TROFF

Commenti

Analogamente a quanto avviene nella ricerca di errori in un programma, il comando TRON attiva un indicatore che stampa ciascuna riga del programma non appena viene eseguita. I numeri vengono visualizzati tra parentesi quadre.

L'esecuzione di TRON può avvenire in modalità diretta o indiretta.

L'indicatore viene disattivato con il comando TROFF, oppure in seguito all'esecuzione del comando NEW.

Esempi

```
TRON
OK
10 K=10
20 FOR J=1 TO 2
30 L=K + 10
40 PRINT J;K;L
50 K=K+10
60 NEXT
70 END
RUN
[10] [20] [30] [40] 1 10 20
[50] [60] [30] [40] 2 20 30
[50] [60] [70]
OK
TROFF
OK
```

UNLOCK (istruzione)

Funzione

Disattiva bloccaggi impostati su un file aperto. Questa forma di protezione viene usata in un ambiente con più unità periferiche, detto normalmente **rete**.

Sintassi

UNLOCK [#]*n*[,[*numerorecord*] [**TO** *numerorecord*]]

Commenti

n è il numero assegnato al file quando è stato aperto.

numerorecord è il numero del record da sbloccare. Nel caso vi fossero più record da sbloccare, devono essere inseriti il numero di record iniziale e quello finale (in questo ordine).

I numeri di record validi vanno da 1 a 2^{32}-1. La dimensione massima di un record è di 32767 byte.

Se non viene specificato il numero del record iniziale, viene usato il valore predefinito 1.

Se non viene specificato il numero del record finale, l'operazione avviene soltanto sul record specificato.

Quelle che seguono sono istruzioni UNLOCK valide:

UNLOCK #*n*	sblocca l'intero file *n*
UNLOCK #*n*,X	sblocca soltanto il record X
UNLOCK #*n*, TO Y	sblocca i record da 1 a Y
UNLOCK #*n*,X TO Y	sblocca i record da X a Y

Prima di poter essere chiusi, tutti i file devono essere sbloccati.

La mancata esecuzione dell'istruzione UNLOCK può complicare i tentativi di accesso a quel file da parte di altri utenti in rete.

Per quanto riguarda i file ad accesso casuale, se viene specificato un intervallo di record, occorre accertarsi che questo intervallo corrisponda a quello precedentemente fornito dall'istruzione LOCK.

Se un'istruzione UNLOCK sintatticamente corretta non può essere eseguita, appare il messaggio `Permesso negato`. L'istruzione UNLOCK deve corrispondere esattamente ad una precedente istruzione LOCK.

Si suppone normalmente che lo spazio di tempo durante il quale un file o un gruppo di record in un file rimangono protetti sia breve, per cui si consiglia di utilizzare la combinazione di istruzioni LOCK/UNLOCK per intervalli di tempo brevi.

Esempi

Quanto segue mostra come le istruzioni LOCK ed UNLOCK dovrebbero essere usate:

```
LOCK #1, 1 TO 4
LOCK #, 5 TO 8
UNLOCK #1, 1 TO 4
UNLOCK #1, 5 TO 8
```

L'esempio seguente non è corretto:

```
LOCK #1, 1 TO 4
LOCK #1, 5 TO 8
UNLOCK #1, 1 TO 8
```

USR (funzione)

Funzione

Chiama una subroutine in linguaggio di assemblaggio.

Sintassi

v=USR[*n*](*argomento*)

Commenti

n specifica la routine da chiamare.

argomento può essere qualsiasi espressione numerica o alfanumerica.

La funzione USR viene usata per il richiamo di routine in linguaggio di assemblaggio. Per questa funzione l'istruzione CALL è comunque da considerarsi più adatta. Per una discussione dettagliata circa il richiamo delle subroutine in linguaggio di assemblaggio, ed un confronto tra CALL e USR, consultare l'appendice D nel *Manuale dell'utente*.

I valori validi per *n* vanno da 0 a 9. Se *n* viene omesso, viene usato il valore predefinito 0 (per le regole di funzionamento di *n*, si veda DEF URS).

L'utilizzo di un segmento diverso da quello predefinito (segmento di dati di GW-BASIC DS), richiede l'esecuzione dell'istruzione DEF SEG prima di una chiamata USR. Ciò assicura che il segmento di codice indichi esattamente la subroutine chiamata.

L'indirizzo del segmento fornito nell'istruzione DEF SEG determina il segmento iniziale della subroutine.

Occorre aver eseguito un'istruzione DEF USR per ciascuna funzione USR, al fine di definire l'offset della chiamata USR. Questo offset, unitamente all'indirizzo del segmento attivo DEG SEG, determina l'indirizzo iniziale della subroutine.

Se sono richieste più di 10 routine dell'utente, si può ridefinire il valore di DEF URS per gli altri indirizzi iniziali quante volte lo si desidera.

Il tipo di variabile (numerica o alfanumerica) che riceve la chiamata di funzione deve essere coerente con l'argomento passato. Se la routine in linguaggio di assemblaggio non richiede alcun argomento, occorre fornire un argomento fittizio.

VAL (funzione)

Funzione

Restituisce il valore numerico della stringa x$.

Sintassi

VAL(x$)

Commenti

La funzione VAL elimina spazi di testa, tabulazioni ed avanzamenti di riga dalla stringa in questione. Ad esempio, la riga seguente restituisce -3:

```
VAL (" -3")
```

La funzione STR$ (per la conversione da valori numerici a stringhe) è complementare alla funzione VAL$.

Se il primo carattere di x$ non è numerico, la funzione VAL(x$) restituisce zero.

Esempi

```
10 READ NOME$,CITTA$,PROV$,CAP$
20 IF VAL(CAP$) <24100 O VAL(CAP$)>24199 THEN
PRINT NOME$ TAB(25) "FUORI PROVINCIA"
30 IF VAL(CAP$)>=24100 AND VAL(CAP$)<=24199 THEN PRINT
NOME$ TAB(25) "BERGAMO"
 .
 .
 .
```

VARPTR (funzione)

Funzione

Restituisce l'indirizzo in memoria della variabile o del Blocco di Controllo File (FCB).

Sintassi

VARPTR(*nomevariabile*)
VARPTR(*#numerofile*)

Commenti

Normalmente, la funzione VARPTR viene usata per ottenere l'indirizzo di una variabile o di una matrice per poi passarla ad una routine in linguaggio d'assemblaggio. Di solito, nel passaggio di una matrice, viene specificata una chiamata di funzione nella forma seguente:

```
VARPTR(A(0))
```

Ciò permette la restituzione dell'elemento con l'indirizzo più basso nella matrice.

Poichè ogni assegnazione di variabile semplice provoca il cambio degli indirizzi contenuti in una matrice, prima di chiamare VARPTR per una matrice, dovrebbero essere assegnate tutte le variabili semplici.

VARPTR (*#numerofile*) restituisce l'indirizzo iniziale del FCB di GW-BASIC assegnato al numero del file.

VARPTR (*nomevariabile*) restituisce l'indirizzo del primo byte di dati identificato con il nome di variabile specificato.

Prima di eseguire VARPTR, bisogna assegnare un valore al nome della variabile; altrimenti, si determina l'errore `Chiamata di funzione illegale`.

Con qualsiasi tipo di variabile (numerica, alfanumerica o matriciale), l'indirizzo restituito rientrerà nell'intervallo tra -32768 e +32767. Se viene restituito un indirizzo negativo, viene addizionato a 65536 per ottenere l'indirizzo effettivo.

Nella tabella seguente vengono mostrati gli offset delle informazioni contenute nel FCB a partire dall'indirizzo restituito da VARPTR:

Tabella 7 Gli offset delle informazioni di FCB

Offset	*Lunghezza*	*Nome*	*Descrizione*
0	1	Mode	La modalità in cui il file è stato aperto: 1 Per solo input 2 Per solo output 4 I/O casuale 16 Per sola aggiunta 32 Uso interno 64 Uso futuro 128 Uso interno
1	38	FCB	Blocco di Controllo File di dischetto.
39	2	CURLOC	Numero dei settori letti o scritti in accesso sequenziale. L'ultimo numero di record +1 per i file ad accesso casuale.
41	1	ORNOFS	Il numero dei byte contenuti in un settore all'atto della lettura o scrittura.
42	1	NMLOFS	Il numero dei byte rimasti nel buffer di INPUT.
43	3	***	Riservato per futura espansione.
46	1	DEVICE	Numero unità periferica: 0-9 Dischi da A: a J: 255 KYBD: 254 SCRN: 253 LPT1: 252 CAS1: 251 COM1: 250 COM2: 249 LPT2: 248 LPT3:
47	1	WIDTH	Larghezza unità.
48	1	POS	Posizione nel buffer di PRINT .

Offset	*Lunghezza*	*Nome*	*Descrizione*
49	1	FLAGS	Uso interno durante BLOAD/BSAVE. Non usato per file di dati.
50	1	OUTPOS	Posizione output usata durante l'espansione di tabulazioni.
51	128	BUFFER	Buffer fisico. Usato per il trasferimento dei dati tra DOS e BASIC. Usare questo offset per l'esame dei dati in modalità I/O sequenziale.
179	2	VRECL	Dimensione record a lunghezza variabile. Il valore predefinito è 128. Impostata dall'opzione di lunghezza in OPEN.
181	2	PHYREC	Numero record fisico corrente.
183	2	LOGREC	Numero record logico corrente
185	1	***	Riservato per futura espansione.
186	2	OUTPOS	Solo per file di disco. Posizione di output per PRINT, INPUT e WRITE.
188	n	FIELD	Memoria creata da FIELD. La dimensione è determinata dal parametro S:. I byte di VRECL vengono trasferiti tra BUFFER e FIELD in operazioni I/O. Usare questo offset per esaminare file in modalità I/O casuale .

Esempio 1

```
100 X=VARPTR(Y)
```

All'esecuzione, X contiene l'indirizzo che indica la posizione di memoria della variabile Y.

Esempio 2

```
10 OPEN "DATA.FIL" AS #1
20 FCBADR = VARPTR(#1)
30 DATADR = FCBADR+1188
40 A$ = PEEK(DATADR)
```

Alla riga 20, FCBADR contiene l'inizio di FCB.

Alla riga 30, DATADR contiene l'indirizzo del buffer dei dati.

Alla riga 40, A$ contiene il primo byte del buffer di dati.

VARPTR$ (funzione)

Funzione

Restituisce l'offset dell'indirizzo di una variabile in memoria sotto forma di un carattere.

Sintassi

VARPTR$(*variabile*)

Commenti

variabile è il nome di una variabile esistente nel programma.

Avvertenza Dal momento che l'assegnazione di nuove variabili semplici provoca il cambiamento degli indirizzi della matrice, prima di chiamare VARPTR$ per un elemento nella matrice, bisogna assegnare tutte le variabili semplici.

VARPTR$ restituisce una stringa di 3 byte nella forma seguente:

```
Byte 0        Byte 1        Byte 2
```

Il byte 0 contiene uno dei tipi di variabile che seguono:

2	intera
3	alfanumerica
4	a precisione singola
8	a precisione doppia

Il byte 1 contiene il formato dell'indirizzo 8086, ed è il byte meno significativo.

Il byte 2 contiene il formato dell'indirizzo 8086, ed è il byte più significativo.

Esempi

```
100 X = USR(VARPTR$(Y))
```

VIEW (istruzione)

Funzione

Definisce il limite fisico di una porta di visualizzazione da *x1,y1* (coordinate dell'angolo superiore sinistro) a *x2,y2* (coordinate dell'angolo inferiore destro).

Sintassi

VIEW [[SCREEN][(*x1,y1*)-(*x2,y2*) [,[*pieno*][,[*bordo*]]]]

Commenti

RUN o VIEW senza argomenti definiscono l'intero schermo come porta di visualizzazione.

(*x1,y1*) sono le coordinate dell'angolo superiore sinistro.

(*x2,y2*) sono le coordinate dell'angolo inferiore destro.

L'attributo *pieno* permette di colorare l'area di visualizzazione.

Se vi è spazio disponibile, l'attributo *bordo* permette di tracciare una riga attorno alla porta di visualizzazione.

Le coordinate *x* e *y* devono rientrare nei limiti dello schermo e devono definire un rettangolo da usarsi per contenere i grafici tracciati. Le coppie di *x* e di *y* vengono ordinate in modo progressivo, partendo dal valore inferiore.

Se viene omesso l'argomento SCREEN, i punti vengono tracciati in relazione alla porta di visualizzazione; cioè, prima che il punto venga tracciato, *x1* e *y1* vengono aggiunte alle coordinate x e y.

Si possono avere tutte le possibili coppie di x e y. L'unico limite è che *x1* non può essere uguale a *x2*, e *y1* non può essere uguale a *y2*.

Se viene incluso l'argomento SCREEN, i punti vengono tracciati in modo assoluto. Sono tracciati solo i punti posti all'interno della porta di visualizzazione.

Quando si usa VIEW, l'istruzione CLS libera soltanto la porta di visualizzazione corrente. Per liberare l'intero schermo, bisogna usare l'istruzione VIEW per disattivare le porte di visualizzazione. Successivamente può essere usato CLS per liberare l'intero schermo. CLS non sposta il cursore ad inizio pagina: a questo fine bisogna premere CTRL-HOME e liberare poi lo schermo.

Esempi

Nella riga seguente viene definita una porta di visualizzazione in modo tale da permettere a PSET(0,0),3 di impostare un punto sullo schermo nella posizione fisica 10,10.

```
VIEW (10,10)-(200,100)
```

Nella riga seguente viene definita una porta in modo tale da impedire la visualizzazione del punto impostato dall'istruzione PSET(0,0),3, in quanto 0,0 non rientra nei limiti della porta stessa. PSET(10,10),3 rientrerebbe in quei limiti.

```
VIEW SCREEN (10,10)-(200,100)
```

VIEW PRINT (istruzione)

Funzione

Imposta i limiti della finestra di testo dello schermo.

Sintassi

VIEW PRINT [*inizio* **TO** *fine*]

Commenti

I parametri *inizio* e *fine* rappresentano la riga iniziale e quella finale della finestra di testo da definire. Se tali parametri non vengono specificati, VIEW PRINT imposta l'intero schermo come finestra per l'inserimento di testo. L'intera area dello schermo è composta dalle righe da 1 a 24; per assunzione, la riga 25 non viene utilizzata.

Le istruzioni e le funzioni che operano all'interno della finestra di testo definita sono CLS, LOCATE, PRINT e SCREEN.

Lo scorrimento dello schermo ed il movimento del cursore sono funzioni utilizzabili unicamente nella finestra di testo.

Per maggiori informazioni si veda VIEW.

WAIT (istruzione)

Funzione

Sospende l'esecuzione del programma e informa circa lo stato della porta di input di un'unità periferica.

Sintassi

WAIT *numeroporta,n*[*,j*]

Commenti

numeroporta è un numero valido di porta di dispositivo che rientra nell'intervallo tra 0 a 65535.

n e *j* sono espressioni intere che rientrano nell'intervallo da 0 a 255.

L'istruzione WAIT determina la sospensione dell'esecuzione fino a che una determinata porta di input sviluppa una certa struttura di bit.

I dati letti nella porta sono XORati con l'espressione intera *j*, e ANDati con *n*.

Se il risultato è zero, GW-BASIC ritorna alla lettura della porta. Se invece il risultato è non zero, l'esecuzione continua con l'istruzione seguente.

Quando eseguita, l'istruzione WAIT esamina il byte *n* per constatare se vi sono bit impostati. Se vi sono bit impostati, il programma continua con l'istruzione seguente anziché attendere che appaia un'intera struttura di bit.

Usando l'istruzione WAIT è possibile l'inserimento di un ciclo infinito. L'uscita da questo ciclo si effettua premendo CTRL-BREAK, o riavviando il sistema.

Il valore predefinito di *j* è zero.

Esempi

```
100 WAIT 32,2
```

Sospende l'attività del dispositivo fino a che la porta 32 riceve 2 come input.

WHILE-WEND (istruzione)

Funzione

Esegue una serie di istruzioni in ciclo fino al verificarsi di un determinato evento.

Sintassi

WHILE *espressione*

.

.

.

[*istruzioni del ciclo*]

.

.

.

WEND

Commenti

Se *espressione* è non zero (vera), le istruzioni all'interno del ciclo vengono eseguite fino ad incontrare l'istruzione WEND. Dopodiché GW-BASIC ritorna all'istruzione WHILE e riesamina l'*espressione*. Se l'*espressione* non ha mutato valore, l'esecuzione continua.

Se il valore è zero (non vero), l'esecuzione viene ripresa invece dall'istruzione che segue WEND.

I cicli di WHILE e WEND possono essere nidificati a qualsiasi livello. Ciascun WEND viene associato al WHILE più recente.

Un'istruzione WHILE senza corrispondente WEND determina l'errore `WHILE senza WEND`. Un'istruzione WEND senza corrispondente WHILE determina l'errore `WEND senza WHILE`.

Esempi

```
90 'ORDINAMENTO RIPETITIVO MATRICE A$
100 FLIPS=1
110 WHILE FLIPS
115 FLIPS=0
120 FOR N=1 TO J-1
130 IF A$(N)>A$(N+1) THEN SWAP A$(N),A$(N+1):FLIPS=1
140 NEXT N
150 WEND
```

WIDTH (istruzione)

Funzione

Imposta la larghezza dello schermo o della stampante in numero di caratteri.

Sintassi

WIDTH *dimensione*
WIDTH *numerofile, dimensione*
WIDTH "*unità*",*dimensione*

Commenti

dimensione è un valore intero tra 0 e 255, e rappresenta la lunghezza impostata.

numerofile è il numero del file aperto.

unità è un'espressione alfanumerica valida che identifica il dispositivo di stampa. I dispositivi validi sono SCRN:, LPT1:, LPT2:, LPT3:, COM1: e COM2:.

Modifica della larghezza dello schermo

Le istruzioni che seguono vengono usate per l'impostazione della larghezza dello schermo. La larghezza permessa è di 40 o 80 colonne.

WIDTH *dimensione*
WIDTH "SCRN:",*dimensione*

Per informazioni più dettagliate, vedere l'istruzione SCREEN.

Il cambio di modalità di schermo cambia la larghezza dello schermo soltanto in caso di passaggio da SCREEN 2 a SCREEN 1 o da SCREEN 1 a SCREEN 0.

Avvertenza Il cambio della larghezza dello schermo ne comporta la liberazione e l'impostazione del bordo in nero.

Modifica della larghezza della stampante

L'istruzione WIDTH che segue viene usata per variare la larghezza della stampante di sistema. Questa istruzione memorizza il valore della nuova larghezza senza effettivamente cambiare l'impostazione corrente della larghezza.

```
WIDTH "LPT1:",dimensione
```

L'istruzione seguente riconosce il valore della larghezza memorizzato:

```
OPEN "LPT1:" FOR OUTPUT AS numero
```

e lo utilizza mentre il file è aperto:

```
WIDTH numerofile,dimensione
```

Con l'apertura del file LPT1:, avviene l'immediato cambiamento della larghezza alla nuova impostazione. Ciò permette di modificare la larghezza come desiderato quando il file è aperto. Questa forma di WIDTH ha significato soltanto per LPT1:. Dopo l'emissione del numero di caratteri indicati dal file aperto, GW-BASIC inserisce un ritorno di riga alla fine della riga, nel caso in cui la larghezza sia inferiore alla lunghezza del record.

Le larghezze valide per la stampante di sistema vanno da 1 a 255.

La specificazione di WIDTH 255 alla stampante di sistema (LPT1:) attiva l'a capo (wrapping) che dà l'impressione di riga infinita.

L'inserimento di qualsiasi valore che non rientra nei limiti specificati determina l'errore `Chiamata di funzione illegale`. Il valore precedente non viene modificato.

L'uso dell'istruzione WIDTH con i file di comunicazioni determina l'impostazione del ritorno di riga alla fine del numero di caratteri specificato dall'attributo *dimensione*. Ciò non ha alcun effetto sul buffer di ricezione o di trasmissione.

Esempi

```
10 WIDTH "LPT1",75
20 OPEN "LPT1:" FOR OUTPUT AS #1
.
.
.
6020 WIDTH #1,40
```

La riga 10 memorizza la larghezza della stampante di sistema in 75 caratteri per riga.

La riga 20 apre il file 1 per la stampante di sistema ed imposta la larghezza a 75 caratteri per l'istruzione PRINT #1 seguente.

La riga 6020 infine, cambia la larghezza corrente della stampante di sistema a 40 caratteri per riga.

WINDOW (istruzione)

Funzione

Traccia righe, grafici ed oggetti in uno spazio non delimitato dai confini dello schermo.

Sintassi

WINDOW[[**SCREEN**](*x1,y1*)-(*x2,y2*)]

Commenti

(*x1,y1*) e (*x2,y2*) sono le coordinate definite dall'utente. Queste coordinate possono essere date da qualsiasi numero a precisione singola e punto mobile. Esse definiscono lo spazio di coordinate che le istruzioni grafiche applicano nello spazio di coordinate fisiche definito da VIEW.

WINDOW è una regione rettangolare nello spazio di coordinate definito e permette zoom e panoramiche. Inoltre, WINDOW permette all'utente di tracciare righe, grafici ed oggetti in un'area non ristretta dai limiti fisici dello schermo. A tal fine, l'utente specifica le combinazioni (*x1,y1*) e (*x2,y2*). Dopodiché, GW-BASIC esegue la conversione della combinazione delle coordinate definite in una combinazione di coordinate fisiche appropriate per eventuali visualizzazioni all'interno dell'area dello schermo.

Se l'attributo SCREEN viene omesso, WINDOW inverte la coordinata *y* nelle istruzioni grafiche che seguono. Questo pone la coordinata (*x1,y1*) nell'angolo inferiore sinistro dello schermo, e la coordinata (*x1,y2*) nell'angolo superiore destro dello schermo. Questo cambio di posizione permette la visualizzazione dello schermo in coordinate cartesiane reali.

Se viene specificato l'attributo SCREEN, l'inversione delle coordinate non viene effettuata. Infatti, le coordinate (*x1,y1*) e (*x2,y2*) vengono rispettivamente poste nell'angolo superiore sinistro e nell'angolo inferiore destro dello schermo.

L'istruzione WINDOW ordina le combinazioni degli argomenti *x* e *y* in ordine crescente.

Ad esempio:

```
WINDOW (50,50)-(10,10)
```

diventa

```
WINDOW (10,10)-(50,50)
```

Oppure

```
WINDOW (-2,2)-(2,-2)
```

diventa

```
WINDOW (-2,-2)-(2,2)
```

Tutte le combinazioni delle coordinate x e y sono valide, ma *x1* e *y1* non possono essere uguali a *x2 e y2*.

WINDOW senza alcun argomento comporta la disattivazione delle precedenti istruzioni.

Esempio 1

Digitando quanto segue:

```
NEW
SCREEN 2
```

lo schermo usa gli attributi standard delle coordinate come segue:

```
0, 0 320, 0 639, 0
\/y crescenti
320,100
```

0, 199 320, 199 639,199

Esempio 2

Digitando quanto segue:

```
WINDOW (-1,-1)-(1,1)
```

lo schermo usa le coordinate cartesiane definite nell'istruzione seguente:

```
-1,1 0,1 1,1

/\y crescenti
0,0
\/y decrescenti
-1,-1 0,-1 1,-1
```

Esempio 3

Digitando quanto segue:

```
WINDOW SCREEN (-1,-1)-(1,1)
```

lo schermo utilizza le coordinate non invertite definite nell'istruzione seguente:

```
-1,-10,-11,-1
/\y decrescente
0,0
\/y crescente
-1,10,11,1
```

Omettendo gli attributi dalle istruzioni RUN, SCREEN e WINDOW si determina la disattivazione di qualsiasi definizione di WINDOW e la conseguente impostazione dello schermo alle coordinate fisiche normali.

WRITE (istruzione)

Funzione

Visualizza l'output dei dati sullo schermo.

Sintassi

WRITE[*elencoespressioni*]

Commenti

Se si omette *elencoespressioni*, viene emessa una riga in bianco. In caso contrario, vengono visualizzati i valori delle espressioni. Le espressioni nell'elenco possono essere numeriche e/o alfanumeriche e devono essere separate da virgole o da punti e virgola.

L'output degli elementi stampati viene visualizzato con l'inclusione di virgole in funzione di separatori. Le stringhe stampate vengono delimitate tramite l'uso di virgolette. Dopo la stampa dell'ultimo elemento, GW-BASIC inserisce un ritorno/avanzamento di riga.

A differenza di PRINT, WRITE inserisce virgole tra gli elementi visualizzati e delimita le stringhe racchiudendole tra virgolette.

I valori numerici vengono emessi nello stesso formato sia da WRITE che da PRINT.

Esempi

```
10 A=80:B=90:C$="QUESTO E' TUTTO"
20 WRITE A,B,C$
RUN
80,90,"QUESTO E' TUTTO"
Ok
```

WRITE# (istruzione)

Funzione

Scrive dati in un file ad accesso sequenziale.

Sintassi

WRITE *#numerofile, elencoespressioni*

Commenti

numerofile è il numero sotto il quale è stata effettuata l'apertura del file per l'output.

elencoespressioni è l'elenco di espressioni alfanumeriche e/o numeriche separate da virgole o punti e virgola.

A differenza dall'istruzione PRINT#, WRITE# inserisce virgole tra gli elementi scritti e delimita le stringhe racchiudendole tra virgolette. Inoltre, WRITE# non inserisce spazi in bianco davanti ai numeri positivi. Dopo la scrittura dell'ultimo elemento, viene inserito un ritorno/avanzamento di riga.

Esempi

Se A$ = "FOTOCOPIATRICE" e B$ = "93604-1". L'istruzione seguente:

```
WRITE#1,A$,B$
```

scrive sul disco quanto segue:

```
"FOTOCOPIATRICE", "93604-1"
```

L'istruzione INPUT$ seguente esegue l'input di "FOTOCOPIATRICE" in A$ e "93604-1" in B$:

```
INPUT#1,A$,B$
```